Optimal Synthesis of Opacity-Enforcing Supervisors for Qualitative and Quantitative Specifications

Yifan Xie, *Student Member, IEEE*, Shaoyuan Li, *Senior Member, IEEE*, and Xiang Yin, *Member, IEEE*

***Abstract*—In this article, we investigate both qualitative and quantitative synthesis of optimal privacy-enforcing supervisors for partially observed discrete-event systems. We consider a dynamic system whose information flow is partially available to an intruder, which is modeled as a passive observer. We assume that the system has a "secret" that does not want to be revealed to the intruder. Our goal is to synthesize a supervisor that controls the system in a *least-restrictive* manner such that the closed-loop system meets the privacy requirement. For the qualitative case, we adopt the notion of *infinite-step opacity* as the privacy specification by requiring that the intruder can never determine for sure that the system is/was at a secret state for any specific instant. If the qualitative synthesis problem is not solvable or the synthesized solution is too restrictive, then we further investigate the quantitative synthesis problem so that the secret is revealed (if unavoidable) *as late as possible* within a finite security-preserving horizon. Effective algorithms are provided to solve both the qualitative and quantitative synthesis problems. Specifically, by building suitable information structures that involve information delays, we show that the optimal qualitative synthesis problem can be solved as a safety game. The optimal quantitative synthesis problem can also be solved as an optimal total-cost control problem over an augmented information structure. Our work provides a complete solution to the standard infinite-step opacity control problem, which has not been solved without an assumption on the relationship between controllable events and observable events. Furthermore, we generalize the opacity enforcement problem to the numerical setting by introducing the secret-revelation-time as a new quantitative measure.**

***Index Terms*—Discrete-event systems (DES), opacity, optimal control, supervisory control.**

Manuscript received 9 February 2023; revised 7 November 2023; accepted 2 December 2023. Date of publication 14 December 2023; date of current version 30 July 2024. This work was supported by the National Natural Science Foundation of China under Grant 62061136004, Grant 61803259, and Grant 61833012. Recommended by Associate Editor J. Komenda. *(Corresponding author: Xiang Yin.)*

Yifan Xie is with the Institute for Systems Theory and Automatic Control, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: yifan.xie@ist.uni-stuttgart.de).

Shaoyuan Li and Xiang Yin are with the Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: syli@sjtu.edu.cn; yinxiang@sjtu.edu.cn).

Digital Object Identifier 10.1109/TAC.2023.3342880

I. Introduction

PRIVACY and security issues have been becoming increasingly more important concerns in cyber-physical systems (CPS) as communications and information exchanges among smart devices may cause information leakage that threatens the system. Formal model-based methods provide rigorous, algorithmic, and correct-by-construction approaches toward the analysis and design of safety critical CPS whose security and privacy demands are ever increasing. In this article, we investigate a formal information-flow security property called *opacity* in the context of discrete-event systems (DES). Roughly speaking, a system is *opaque* if its "secret" can never be revealed to a malicious intruder that can access the information flow of the system. The notion of opacity is essentially a confidentiality property that captures the plausible deniability for the system's secrets [1].

In the context of DES, opacity has been studied very extensively in the past few years, see, e.g., some recent works [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], and the survey papers [16] and [17]. Especially, to characterize different types of security requirements, different notions of opacity are proposed in the literature. For example, current-state opacity (respectively, initial-state opacity) requires that the intruder should never know for sure that the system is currently at (respectively, was initially from) a secret state by utilizing the information available up to the current instant [18]. In many situations, the intruder may further use future information to better infer the security status of the system for some previous instants, which is essentially an information smoothing process. To capture this scenario, the notions of K-step opacity [19], [20] and infinite-step opacity [19], [21], [22] are proposed. Particularly, infinite-step opacity is the strongest one among all notions of opacity mentioned above, which requires that the intruder can never know for sure that the system is/was at a secret state for any specific instant even by using future information.

When a system is verified to be nonopaque, one important problem is to enforce opacity via some mechanisms. In general, there are two approaches for enforcing opacity: one is to control the actual behavior of the system so that those secret-revealing behaviors can be avoided [23], [24], [25], [26], [27], [28], [29], [30], [31] and the other one is to change the information flow of the system so that the intruder can be "cheated" or be "confused"

[32], [33], [34], [35], [36], [37], [38], [39], [40]. In particular, the first approach is essentially the supervisory control of opacity that aims to find a *supervisor* that restricts the behavior of the system dynamically such that the closed-loop system is opaque. For example, Dubreil et al. [30] studied the supervisory control problem for current-state opacity assuming that all controllable events are observable and the intruder cannot observe more events than the supervisor. Tong et al. [29] relaxed the above assumptions but assumed that the intruder does not know the implementation of the supervisor. In [41], nondeterministic supervisors are used to enhance the plausible deniability of the controlled system. Note that all the above mentioned works on opacity-enforcing supervisory control consider current-state opacity.

In this article, we study the problem of synthesizing optimal supervisors for *infinite-step opacity* and its quantitative generalization. Enforcing infinite-step opacity is significantly more challenging than the standard current-state opacity enforcement problem. Specifically, in the infinite-step opacity setting, whether or not a secret can be revealed to the intruder not only depends on the information available currently, but also depends on the information in the future. To handle this future information, in the verification problem, we can look ahead in the original open-loop system by "borrowing" the future information from the fixed plant model, see, e.g., [19] and [21]. However, in the synthesis problem, the future information of the closed-loop system is *unknown* and depends on the control policy in the future, which is *to be determined*. Hence, how to handle the dependency between the delayed information and the future control policy is the main difficulty in the synthesis of infinite-step opacity. In this article, we propose effective approaches that solve both the qualitative and quantitative versions of the infinite-step opacity control synthesis problem. Our main contributions are twofold.

1) First, we provide a new synthesis algorithm that solves the standard qualitative infinite-step opacity supervisory control problem without any assumption on the relationship between controllable events and observable events. Our approach is based on the generic structure of bipartite transition systems (BTSs) [28] and a new type of information state that effectively captures the issue of information delay while avoiding the dependence on future control policy. In particular, we show that the proposed synthesis algorithm is sound and complete, and the resulting supervisor is maximally permissive in terms of language inclusion. Therefore, for the qualitative part, we completely solve the standard infinite-step opacity control problem, which was only partially solved in the literature under restrictive assumptions.
2) Furthermore, we investigate a quantitative version of the opacity-enforcing control problem by introducing a secret revelation cost based on the notion of *secret-revelation-time*, i.e., the earlier the secret is revealed, the higher the cost will be. This leads to a numerical generalization of infinite-step opacity (as well as the notion of K-step opacity [19], [20]). The control objective is to reveal each visit of secret states (if unavoidable) *as late as possible*. By suitably augmenting timing information into the information state space, we show that this problem can be solved effectively as an optimal worst-case total-cost control problem. Our approach provides a new angle for quantifying secret in control synthesis with information delays using the concept of secret-revelation-time.

Our work is also related to several works in the literature. Regarding the qualitative synthesis problem, as we mentioned earlier, most of the existing works on opacity-enforcing supervisory control only consider current-state opacity. One exception is [42], where Saboori and Hadjicostis proposed a method to enforce infinite-step opacity by synthesizing *a set of supervisors* that run synchronously. However, our approach synthesizes a single supervisor directly. More importantly, the approach in [42] is based on the restrictive assumption that all controllable events are observable; this assumption is also relaxed in our approach. In our recent work [35], we consider the synthesis of dynamic masks for infinite-step opacity. However, dynamic masks can only change the observation of the system, while supervisor can change the *actual behavior* of the system without interfering the information flow directly. Hence, different information-state updating rules are proposed here to handle the control problem. In terms of quantification of opacity, most of the existing works focus on qualifying *how opaque* the system is in terms of probability measure, see, e.g., [2], [5], [43], [44], [45], and [46]. In the very recent work [47], Lefebvre and Hadjicostis proposed the concept of opacity revelation time to characterize how long the initial secret is kept. This concept is closely related to our secret-revelation-time. However, here we essentially consider a delayed-state-estimation problem for each visit of secret states, which is much more involved than the initial-state-estimation problem considered in [47]. Furthermore, we consider the control synthesis problem, while the work in [47] only considered the verification problem.

The rest of this article is organized as follows. Section II presents some necessary preliminaries. In Section III, we formulate the qualitative opacity-enforcing control problem. In Section IV, we present a new class of information states for infinite-step opacity. In Section V, we first define BTSs and then present a synthesis algorithm that returns a maximally permissive partial-observation supervisor to enforce infinite-step opacity. Furthermore, we quantify infinite-step opacity via secret-revelation-time and solve the quantitative synthesis problem in Section VI. Finally, Section VII concludes this article. Some preliminary results for the qualitative part were presented in [48] without proof. This article presents complete proofs as well as detailed explanations. Moreover, we further investigate the quantitative synthesis problem by quantifying the secret-revelation-time. The techniques for solving the qualitative and quantitative problems are also different.

II. Preliminary

A. System Model

We assume basic knowledge of DES and use common notations, see, e.g., [49]. Let Σ be an alphabet. A string is a finite sequence of events $s = \sigma_1 \dots \sigma_n, \sigma_i \in \Sigma$ for all $i = 1, \dots, n$; $|s| = n$ denotes its length. We denote by Σ^* the set of all strings over Σ including the empty string ϵ whose length is

zero. A language $L \subseteq \Sigma^*$ is a set of strings, and we define $\overline{L} = \{u \in \Sigma^* : \exists v \in \Sigma^*, uv \in L\}$ as the prefix closure of L. For the sake of simplicity, we write $s \leq t$ if $s \in \overline{\{t\}}$; we write $s < t$ if $s \leq t$ and $s \neq t$.

A DES is modeled as a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$, where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, where $\delta(x, \sigma) = y$ means that there is a transition labeled by event σ from state x to y, and $x_0 \in X$ is the initial state. The transition function can also be extended to $\delta : X \times \Sigma^* \to X$ in the usual manner [49]. For simplicity, we write $\delta(x, s)$ as $\delta(s)$ when $x = x_0$. The language generated by G is defined by $\mathcal{L}(G) := \{s \in \Sigma^* : \delta(s)!\}$, where ! means "is defined".

When the system is partially observed, Σ is partitioned into two disjoint sets: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined recursively by

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma, & \text{if } \sigma \in \Sigma_o \\ P(s), & \text{if } \sigma \in \Sigma_{uo}. \end{cases}$$

We also extend the natural projection to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ by: for any $L \subseteq \Sigma^*$, we have $P(L) = \{P(s) \in \Sigma_o^* : s \in L\}$. For any observation $\alpha\beta \in P(\mathcal{L}(G))$, we define $\hat{X}_G(\alpha \mid \alpha\beta)$ as the *delayed-state estimate* that captures the set of all possible states the system could be in at the instant when α is observed given the entire observation $\alpha\beta$, i.e.,

$$\hat{X}_G(\alpha \mid \alpha\beta) := \{\delta(s) \in X : st \in \mathcal{L}(G), P(s) = \alpha, P(st) = \alpha\beta\}.$$

For simplicity, we define $\hat{X}_G(\alpha) := \hat{X}_G(\alpha \mid \alpha)$ as the current-state estimate upon the occurrence of α.

B. Secret and Intruder

We assume that the privacy specification of the system is captured by a set of secret states $X_S \subseteq X$. We consider an intruder modeled as a passive observer having the following capabilities:

A1 The intruder knows the system model G;

A2 The intruder can observe the occurrence of each observable event in Σ_o.

Then, the intruder can infer whether or not the system is/was at a secret state based on the information flow available. Specifically, we use the notion of *infinite-step opacity* to describe the privacy requirement of the system, which says that for any string that leads to a secret state, the intruder can never determine for sure whether the system is/was at a secret state for any current/previous instant.

Definition 1: Given system G, a set of observable events Σ_o, and a set of secret states X_S, system G is said to be infinite-step opaque w.r.t. X_S and Σ_o if

$$\forall \alpha\beta \in P(\mathcal{L}(G)), \hat{X}_G(\alpha \mid \alpha\beta) \not\subseteq X_S.$$

The following example illustrates infinite-step opacity.

Example 1: Let us consider system G in Fig. 1(a) with $\Sigma_o = \{o_1, o_2\}$ and $X_S = \{5\}$. Then, G is not infinite-step opaque w.r.t. X_S and Σ_o. To see this, we consider string abo_1o_2 with $P(abo_1o_2) = o_1o_2$, then the delayed-state estimate is $\hat{X}_G(o_1 \mid o_1o_2) = \{5\} \subseteq X_S$. This means that the intruder knows for sure that the system was at a secret state one step ago when string o_1o_2 is observed. Upon the occurrence of o_1o_2, the secret state 5 will be revealed to the intruder.

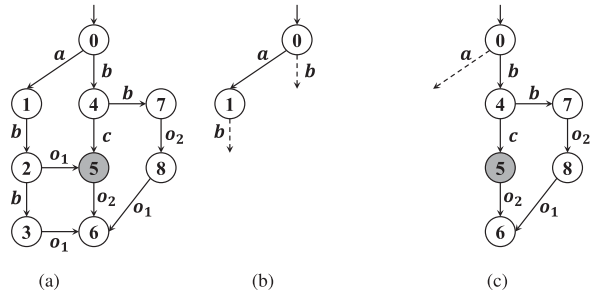


Fig. 1. For G: $\Sigma_o = \{o_1, o_2\}, \Sigma_c = \{a, b, c\}$, and $X_S = \{5\}$. (a) System G. (b) System G_1. (c) System G_2.

III. Qualitative Privacy-Enforcing Control Problem

We start by the qualitative synthesis problem for infinite-step opacity. When a given system G is not opaque, i.e., the secret of the system can be revealed to the intruder, we need to enforce opacity on the system. One typical approach is to synthesize a *supervisor* that restricts the system's behavior to a sublanguage that satisfies the privacy requirement.

In the framework of supervisory control, a supervisor can restrict the behavior of G by dynamically enabling/disabling some events. In this setting, we assume that the events set is further partitioned as $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. Controllable events are events that can be prevented from happening, or disabled by the supervisor; uncontrollable events cannot be disabled by the supervisor. A control decision $\gamma \in 2^\Sigma$ is said to be admissable if $\Sigma_{uc} \subseteq \gamma$, namely, uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissable control decisions or control patterns. Since a supervisor can only make decisions based on its observation, a partial-observation supervisor is a function

$$S : P(\mathcal{L}(G)) \to \Gamma.$$

We use notation S/G to represent the closed-loop system under supervision and the language generated by S/G, denoted by $\mathcal{L}(S/G)$, is defined recursively by the following manners:

1) $\epsilon \in \mathcal{L}(S/G)$;
2) for any $s \in \Sigma^*, \sigma \in \Sigma$, we have $s\sigma \in \mathcal{L}(S/G)$ iff $s\sigma \in \mathcal{L}(G), s \in \mathcal{L}(S/G)$, and $\sigma \in S(P(s))$.

In this article, we want to synthesize a supervisor that disables events dynamically based on the observation trajectory such that the controlled system S/G satisfies the privacy requirement, e.g., being infinite-step opaque for the qualitative setting. Note that the implementation of supervisor S may become a public information. Therefore, we further make the following assumption:

A3 The intruder knows the functionality of the supervisor S.

Then, under assumption A3, the intruder may further know that some behaviors in the original open-loop system are no longer feasible in the closed-loop system. Therefore, for any

observable string $\alpha\beta \in P(\mathcal{L}(S/G))$, when the system is controlled by supervisor S, we define

$$\hat{X}_{S/G}(\alpha \mid \alpha\beta) := \left\{ \delta(s) \in X : \begin{array}{c} st \in \mathcal{L}(S/G), \\ P(s) = \alpha, P(st) = \alpha\beta \end{array} \right\}$$

as the delayed-state estimate in the closed-loop system. Similarly, we define $\hat{X}_{S/G}(\alpha) = \hat{X}_{S/G}(\alpha \mid \alpha)$. Then, we say that the closed-loop system S/G is infinite-step opaque w.r.t. X_S and Σ_o if

$$\forall \alpha\beta \in P(\mathcal{L}(S/G)), \hat{X}_{S/G}(\alpha \mid \alpha\beta) \not\subseteq X_S. \tag{1}$$

In the supervisory control framework, it is desirable that the supervisor enables as many events as possible to leave autonomy for the original system. Therefore, our goal is to find a maximally permissive supervisor (in terms of language inclusion). Then, the qualitative privacy-enforcing supervisory control problem for infinite-step opacity is formulated as follows.

Problem 1 (Qualitative opacity-enforcing control problem): Given system G and a set of secret states $X_S \subseteq X$, synthesize a partial-observation supervisor $S : P(\mathcal{L}(G)) \to \Gamma$, such that

(i) S/G is infinite-step opaque w.r.t. X_S and Σ_o;

(ii) for any other supervisor S' satisfying (i), we have $\mathcal{L}(S/G) \not\subset \mathcal{L}(S'/G)$.

The second condition implies that the synthesized supervisor is maximal in the sense that its permissiveness cannot be improved anymore. As we will see in the following example, such a maximal solution is not unique in general.

Example 2: To enforce infinite-step opacity for system G, as shown in Fig. 1(a), we need to find a supervisor such that the closed-loop behavior of the controlled system is opaque. Let $\Sigma_c = \{a, b, c\}$ be the set of controllable events. By disabling event b initially, we can get an infinite-step opaque system, as shown in Fig. 1(b). By disabling event a initially, we can get another infinite-step opaque system, as shown in Fig. 1(c). One can easily check that both $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$ satisfy condition (ii) in Problem 1. It is worth noting that both of them are infinite-step opaque and maximal, i.e., they satisfy condition (i) and (ii) in Problem 1. However, the union of $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$ is not a feasible solution since the supervisor needs to disable b initially in $\mathcal{L}(G_1)$ but needs to enable b initially in $\mathcal{L}(G_2)$.

Remark 1: In Problem 1, the control objectives are infinite-step opacity and maximally permissiveness. In some problems, one may further require that the closed-loop system is *nonblocking* with respect to a given set of marked states. We do not tackle the issue of nonblockingness in this work since it requires completely different techniques. Interested readers are referred to [50] for how to solve the nonblocking supervisory control problem under partial observation. In principle, the techniques in [50] can be combined with our results in this work to ensure nonblockingness in addition to the opacity requirement.

Finally, we introduce some necessary operators that will be used for further developments. Let $q \in 2^X$ be a set of states, $\gamma \in \Gamma$ be a control decision, and $\sigma \in \Sigma_o$ be an observable event. The *unobservable reach* of $q \subseteq X$ under control decision $\gamma \subseteq \Sigma$ is defined by

$$\mathrm{UR}_\gamma(q) := \{\delta(x, w) \in X : x \in q, w \in (\Sigma_{uo} \cap \gamma)^*\}, \tag{2}$$

which is the set of states that can be reached from states in q via unobservable strings allowed in γ. The observable reach of $q \subseteq X$ upon the occurrence of $\sigma \in \Sigma_o$ is defined by

$$\mathrm{NX}_\sigma(q) := \{\delta(x, \sigma) \in X : x \in q\}, \tag{3}$$

which is the set of states that can be reached from states in q upon the occurrence of the observable event $\sigma \in \Sigma_o$. Similarly, let $\rho \in 2^{X \times X}$ be a set of state pairs. Intuitively, for each $(x, x') \in \rho$, x represents where the system was from at some instants, and x' represents where the system is currently at. Let $\gamma \in \Gamma$ be a control decision and $\sigma \in \Sigma_o$ be an observable event. We define

$$\widetilde{\mathrm{UR}}_\gamma(\rho) = \{(x, \delta(x', w)) \in X \times X : (x, x') \in \rho, w \in (\Sigma_{uo} \cap \gamma)^*\}, \tag{4}$$

$$\widetilde{\mathrm{NX}}_\sigma(\rho) = \{(x, \delta(x', \sigma)) \in X \times X : (x, x') \in \rho\}, \tag{5}$$

$$\odot_\gamma(q) = \{(x, \delta(x, w)) \in X \times X : x \in q, w \in (\Sigma_{uo} \cap \gamma)^*\}. \tag{6}$$

Intuitively, $\widetilde{\mathrm{UR}}_\gamma(\rho)$ and $\widetilde{\mathrm{NX}}_\sigma(\rho)$, respectively, modify the unobservable reach and the observable reach, by not only tracking all possible current states, but also tracking where they come from. Also, $\odot_\gamma(q)$ maps q to a set of state pairs such that, in each pair of states, the first state can reach the second state via enabled but unobservable strings.

Example 3: Let us consider system G in Fig. 1(a) again. Let $q = \{1, 5\} \in 2^X$ be a set of states and $\gamma = \{o_1, o_2, a, b\} \in \Gamma$ be a control decision. Then, we have $\mathrm{UR}_\gamma(\{1, 5\}) = \{1, 2, 3, 5\}$. Upon the occurrence of enabled observable event $o_1 \in \Sigma_o \cap \gamma$, we know that $\mathrm{NX}_{o_1}(\{1, 2, 3, 5\}) = \{5, 6\}$. Let $\rho = \{(0, 1), (4, 5)\} \in 2^{X \times X}$ be a set of state pairs, which represents that the system is either currently at state 1 from state 0; or currently at state 5 from state 4. Let $\gamma = \{o_1, o_2, a, b\} \in \Gamma$ be a control decision. Then, we have $\widetilde{\mathrm{UR}}_{\{o_1, o_2, a, b\}}(\{(0, 1), (4, 5)\}) = \{(0, 1), (0, 2), (0, 3), (4, 5)\}$. Upon the occurrence of enabled observable event $o_1 \in \Sigma_o \cap \gamma$, we have $\widetilde{\mathrm{NX}}_{o_1}(\{(0, 1), (0, 2), (0, 3), (4, 5)\}) = \{(0, 5), (0, 6)\}$. Besides, let $q = \{1, 2, 3, 5\} \in 2^X$ and $\gamma = \{o_1, o_1, a, b\}$, we have $\odot_\gamma(q) = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3), (5, 5)\}$.

IV. Information State and Its Flow

To enforce infinite-step opacity, the main difficulty is that the state estimation is *delayed* and one can use future observation to improve its knowledge about the system at some previous instants. In this section, we study how information evolves in the closed-loop system when delayed information is involved.

A. Notion of Information State

In the synthesis of partially observed systems, 2^X is usually chosen as the set of *information states* representing current information of the system. This information state has been shown to be suitable for current-state opacity enforcement. However, infinite-step opacity requires that the secret cannot be revealed at any instant currently and in the future; hence, 2^X is not sufficient enough.

Note that, the requirement of infinite-step opacity in the following can be equivalently written as:

$$\forall \alpha \in P(\mathcal{L}(S/G)) \quad \forall \alpha' \leq \alpha : \hat{X}_{S/G}(\alpha' \mid \alpha) \not\subseteq X_S. \tag{7}$$

This reformulation suggests that, to enforce infinite-step opacity, it is sufficient to capture the delayed-state estimates of all previous instants. Therefore, instead of using 2^X, we use another set of information states defined by

$$I := 2^X \times 2^{2^{X\times X}}.$$

Then, each information state $\imath \in I$ is in the form of $\imath = (C(\imath), D(\imath))$, where the following holds.

1) The first component $C(\imath) \in 2^X$ is a set of states that captures the current-state estimate of the system.
2) The second component is $D(\imath) \in 2^{2^{X\times X}}$, whose elements are sets of state pairs that captures all possible delayed-state estimates in history. Specifically, each element $\rho \in D(\imath)$ is in the form of $\{(x_1, x_1'), \ldots, (x_k, x_k')\}$, where $x_i \in X$ represents the state of the system at some previous instants and $x_i' \in X$ represents the current state of the system for all $i = 1, \ldots, k$, and ρ contains all possibilities for that instant. Then, $D(\imath) = \{\rho_1, \ldots, \rho_n\}$ essentially contains all such sets of pairs for all previous instants.

The intuitions for defining the above information structure are as follows. For the infinite-step opacity synthesis problem, according to (7), we need to consider the effects of current control decision *for all previous instants*. To this end, we need to keep track of all delayed-state estimates for all previous instants as well as the current-state estimate. This is why each information state consists of two parts: the first part corresponds to the current-state estimate whose domain is 2^X and the second part corresponds to all possible delayed-state estimates whose domain is $2^{2^{X\times X}}$. Without these information, one cannot check whether or not some previous secrets have been released or not.

B. Information State Updating Rule

Suppose that the system's information state is $\imath = (C(\imath), D(\imath))$. Then, upon the occurrence of an observable event $\sigma \in \Sigma_o$ (should also be allowed by the previous control decision), the supervisor will issue a new control decision $\gamma \in \Gamma$. Therefore, (σ, γ) is our new information about the system and the information state $\imath = (C(\imath), D(\imath))$ needs to be updated to $\imath' = (C(\imath'), D(\imath'))$ as follows:

Information state updating rule

$$\begin{cases} C(\imath') = \mathrm{UR}_\gamma(\mathrm{NX}_\sigma(C(\imath))) \\ D(\imath') = \{\widetilde{\mathrm{UR}}_\gamma(\widetilde{\mathrm{NX}}_\sigma(\rho)) \in 2^{X\times X} : \rho \in D(\imath)\} \\ \qquad \cup\{\odot_\gamma(C(\imath'))\}. \end{cases} \tag{8}$$

Intuitively, the first equation simply updates the current-state estimate of the system, while the second equation updates all possible delayed-state estimates (pairs) of previous instants and adds the current-state estimate to the history. Then, we consider a controlled system S/G. Let $\alpha = \sigma_1\sigma_2\ldots\sigma_n \in P(\mathcal{L}(S/G))$ be an observable string. Then, the following information states evolution will happen:

$$\imath_0 \xrightarrow{(\sigma_1, S(\sigma_1))} \imath_1 \xrightarrow{(\sigma_2, S(\sigma_1\sigma_2))} \cdots \xrightarrow{(\sigma_n, S(\sigma_1\ldots\sigma_n))} \imath_n \tag{9}$$

where $\imath_0 = (\mathrm{UR}_{S(\epsilon)}(\{x_0\}), \{\odot_{S(\epsilon)}(\mathrm{UR}_{S(\epsilon)}(\{x_0\}))\})$ represents the initial information state, and each $\imath_{i-1} \xrightarrow{(\sigma_i, S(\sigma_1\ldots\sigma_i))} \imath_i$ means that $\imath_i$ is obtained from $\imath_{i-1}$ with new information $(\sigma_i, S(\sigma_1\ldots\sigma_i))$ according to the updating rule in (8). We denote by $\mathcal{I}(\alpha)$ the information state reached by α, i.e., $\mathcal{I}(\alpha) = \imath_n$.

We illustrate the above information updating procedure by the following example.

Example 4: Let S be the supervisor that results in closed-loop system G_2 in Fig. 1(c). Specifically, supervisor S disables a initially and then enables all events, i.e., $S(\epsilon) = \{o_1, o_2, b, c\}$ and $S(\alpha) = \Sigma$ for $\alpha \neq \epsilon$. Let us consider observable string $o_2o_1 \in P(\mathcal{L}(S/G))$. Then, we have

$$\mathcal{I}(\epsilon) = (\mathrm{UR}_{S(\epsilon)}(\{x_0\}), \{\odot_{S(\epsilon)}(\mathrm{UR}_{S(\epsilon)}(\{x_0\}))\}),$$

where

$$C(\mathcal{I}(\epsilon)) = \{0, 4, 5, 7\},$$
$$D(\mathcal{I}(\epsilon)) = \left\{\!\!\left\{ \begin{matrix} (0,0),(0,4),(0,5),(0,7), \\ (4,4),(4,5),(4,7),(5,5),(7,7) \end{matrix} \right\}\!\!\right\}.$$

Once event o_2 is observed, new control decision $S(o_2) = \{o_1, o_2, a, b, c\}$ is made, and the updated information state is $\mathcal{I}(o_2)$, where

$$\begin{aligned} C(\mathcal{I}(o_2)) &= \mathrm{UR}_{S(o_2)}(\mathrm{NX}_{o_2}(C(\mathcal{I}(\epsilon)))) \\ &= \{6, 8\}, \\ D(\mathcal{I}(o_2)) &= \{\widetilde{\mathrm{UR}}_{S(o_2)}(\widetilde{\mathrm{NX}}_{o_2}(\rho)) : \rho \in D(\mathcal{I}(\epsilon))\} \\ &\quad \cup \{\odot_{S(o_2)}(C(\mathcal{I}(o_2))\} \\ &= \left\{\!\!\left\{ \begin{matrix} \{(0,6),(0,8),(4,6),(4,8),(5,6),(7,8)\}, \\ \{(6,6),(8,8)\} \end{matrix} \right\}\!\!\right\}. \end{aligned}$$

Then, event o_1 is observed and new control decision $S(o_2o_1) = \{o_1, o_2, a, b, c\}$ is made. The information state is then updated to $\mathcal{I}(o_2o_1)$, where

$$\begin{aligned} C(\mathcal{I}(o_2o_1)) &= \mathrm{UR}_{S(o_2o_1)}(\mathrm{NX}_{o_1}(C(\mathcal{I}(o_2)))) \\ &= \{6\}, \\ D(\mathcal{I}(o_2o_1)) &= \{\widetilde{\mathrm{UR}}_{S(o_2o_1)}(\widetilde{\mathrm{NX}}_{o_1}(\rho)) : \rho \in D(\mathcal{I}(o_2))\} \\ &\quad \cup \{\odot_{S(o_2o_1)}(C(\mathcal{I}(o_2o_1))\} \\ &= \{\{(0,6),(4,6),(7,6)\}, \{(8,6)\}, \{(6,6)\}\}. \end{aligned}$$

C. Property of the Information State

In the following result, we formally show that the proposed information state updating rule indeed yields the desired delayed-state estimate in the controlled system.

Proposition 1: Let S be a supervisor, $\alpha \in P(\mathcal{L}(S/G))$ be an observable string, and $\mathcal{I}(\alpha)$ be the information state reached. Then, we have the following properties:

i) $C(\mathcal{I}(\alpha)) = \hat{X}_{S/G}(\alpha)$

ii) $D(\mathcal{I}(\alpha)) = \{\rho_{\alpha',\alpha} \in 2^{X\times X} : \alpha' \leq \alpha\}$, where

$$\rho_{\alpha',\alpha} = \left\{ (\delta(s),\delta(st)) \in X \times X : \begin{array}{c} st \in \mathcal{L}(S/G), \\ P(s) = \alpha', P(st) = \alpha \end{array} \right\}. \tag{10}$$

Recall that each information state is in the form of $\imath = (C(\imath), D(\imath)) \in 2^X \times 2^{2^{X\times X}}$, where $D(\imath)$ is a set of possible state pairs. We define

$$D_1(\imath) := \{\{x \in X : (x,x') \in \rho\} : \rho \in D(\imath)\} \tag{11}$$

as the set of its first components. For example, for $\mathcal{I}(o_2) = (\{6,8\},\{\{(0,6),(0,8),(4,6),(4,8),(5,6),(7,8)\},\{(6,6),(8,8)\}\})$ in Example 4, we have $D_1(\mathcal{I}(o_2)) = \{\{0,4,5,7\},\{6,8\}\}$. The following result shows that $D_1(\imath)$ indeed captures all possible delayed-state estimates.

Corollary 1: Let S be a supervisor, $\alpha \in P(\mathcal{L}(S/G))$ be an observable string, and $\mathcal{I}(\alpha)$ be the information state reached. Then, we have

$$D_1(\mathcal{I}(\alpha)) = \{\hat{X}_{S/G}(\alpha' \mid \alpha) \in 2^X : \alpha' \leq \alpha\}.$$

Proof: By Proposition 1, we know that $D(\mathcal{I}(\alpha)) = \{\rho_{\alpha',\alpha} \in 2^{X\times X} : \alpha' \leq \alpha\}$. Therefore

$$\begin{aligned} &D_1(\mathcal{I}(\alpha)) \\ &= \{\{x \in X : (x,x') \in \rho_{\alpha',\alpha}\} : \alpha' \leq \alpha\} \\ &= \left\{ \left\{ \delta(s) \in X : \begin{array}{c} st \in \mathcal{L}(S/G), \\ P(s) = \alpha', P(st) = \alpha \end{array} \right\} : \alpha' \leq \alpha \right\} \\ &= \{\hat{X}_{S/G}(\alpha' \mid \alpha) \in 2^X : \alpha' \leq \alpha\}. \end{aligned}$$

■

V. Solving the Qualitative Synthesis Problem

In this section, we discuss how to synthesize a maximally permissive supervisor that enforces infinite-step opacity. Our approach is to first construct a structure called the BTS that contains all opaque solutions and then extract a maximally permissive supervisor from it. The existence of such a BTS can also be used to verify whether or not there exists a supervisor such that the closed-loop system is infinite-step opaque.

A. Bipartite Transition System

In Section IV, we have proposed a new type of information state that can capture all possible delayed-state estimates. Note that, the information state updating rule as defined in (8) essentially consists of two steps: the immediate observable reach when a new observable event occurs and the unobservable reach when a new control decision is issued. However, this is based on the assumption that supervisor S is given. In the synthesis problem, the control decision at each instant is unknown and to be determined. Therefore, we need to separate these two updating steps clearly. To this end, we adopt the generic structure of the BTS that was originally proposed in [28] by incorporating the new information state.

Definition 2: A BTS T w.r.t. G is a 7-tuple

$$T = \left(Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma_o, \Gamma, y_0^T\right), \tag{12}$$

where the following holds:

1) $Q_Y^T \subseteq I$ is the set of Y-states. Therefore, a Y-state $y \in Q_Y^T$ is in the form of $y = (C(y), D(y))$;
2) $Q_Z^T \subseteq I \times \Gamma$ is the set of Z-states. For each $z \in Q_Z^T$, and $I(z)$ and $\Gamma(z)$ denote, respectively, the information state and the control decision, so that $z = (I(z), \Gamma(z)) \in 2^X \times 2^{2^{X\times X}} \times \Gamma$. For simplicity, we further write $z = (C(I(z)), D(I(z)), \Gamma(z))$ as $z = (C(z), D(z), \Gamma(z))$;
3) $h_{YZ}^T : Q_Y^T \times \Gamma \to Q_Z^T$ is the partial transition function from Y-states to Z-states satisfying the following constraint: for any $h_{YZ}^T(y,\gamma) = z$, we have

$$\begin{cases} C(z) = \mathrm{UR}_\gamma(C(y)) \\ D(z) = \left\{ \widetilde{\mathrm{UR}}_\gamma(\rho) \in 2^{X\times X} : \rho \in D(y) \right\} \cup \left\{ \odot_\gamma(C(z)) \right\} \\ \Gamma(z) = \gamma \end{cases} \tag{13}$$

4) $h_{ZY}^T : Q_Z^T \times \Sigma \to Q_Y^T$ is the partial transition function from Z-states to Y-states satisfying the following constraint: for any $h_{ZY}^T(z,\sigma) = y$, we have $\sigma \in \Gamma(z) \cap \Sigma_o$ and

$$\begin{cases} C(y) = \mathrm{NX}_\sigma(C(z)) \\ D(y) = \left\{ \widetilde{\mathrm{NX}}_\sigma(\rho) \in 2^{X\times X} : \rho \in D(z) \right\} \end{cases} \tag{14}$$

5) Σ_o is the set of observable events of G;
6) Γ is the set of admissible control decisions of G;
7) $y_0^T := (\{x_0\}, \{\emptyset\}) \in Q_Y^T$ is the initial Y-state.

The BTS is essentially a game structure between the controller and the environment. When the controller picks a control decision $\gamma \in 2^\Sigma$ at a Y-state, the game moves to a Z-state. A transition from Y-state to Z-state is an unobservable reach under the issued control decision and remembers the control decision. When the environment picks an observation $\sigma \in \Sigma_o \cap \gamma$ at a Z-state, the game moves to a Y-state, and so forth. A transition from Z-state to Y-state is the observable reach. Transitions from Z-states to Y-states and transitions from Y-states to Z-states are indeed the information state updating rule in (8), but we separate the updating procedure into two parts. Specifically, for any $z \in Q_Z^T, y \in Q_Y^T, \gamma \in \Gamma$, and $\sigma \in \Sigma_o$, we have $h_{YZ}^T(h_{ZY}^T(z,\sigma),\gamma) = (\imath',\gamma)$, where $I(z) \xrightarrow{(\sigma,\gamma)} \imath'$. For simplicity, we write a transition as h_{YZ} whenever it is defined for some h_{YZ}^T, and the same for h_{ZY}. The basic generic structure of the BTS was originally proposed in [28]. Here, we generalize the original BTS by using new information states capturing delays rather than the current-state-type information states used in [28].

B. Supervisor Synthesis Algorithm

By (7) and Corollary 1, to make sure that the closed-loop system S/G is infinite-step opaque, it suffices to guarantee that, for any information state $\imath$ reached, we have $\forall q \in D_1(\imath), q \nsubseteq X_S$. Therefore, we define

$$Q_{\text{rev}} = \{z \in Q_Z : q \in D_1(I(z)), q \subseteq X_S\}$$

as the set of *secret-revealing* Z-states. To synthesize a supervisor that enforces infinite-step opacity, the controlled system S/G needs to guarantee that all reachable information states are not secret revealing. Furthermore, as the supervisor can only play at Y-states, we need to make sure that 1) there is at least one choice at each Y-state, and 2) all choices at each Z-state should be considered. Therefore, for each BTS T, we say that a state is *consistent* if the following holds.

1) At least one transition is defined when it is a Y-state.
2) All feasible observations are defined when it is a Z-state.

Let $\mathcal{T}$ be the set of all BTSs. For any BTS $T \in \mathcal{T}$, we define Q^T_{consist} as the set of consistent states in T. Also, for a set of states $Q \subseteq Q^T_Y \cup Q^T_Z$, we define $T|_Q$ as the restriction of T to Q, i.e., $T|_Q$ is the BTS obtained by removing states not in Q and their associated transitions from T.

In order to synthesize an opacity-enforcing supervisor, first, we construct the largest BTS that enumerates all possible transitions for each state, i.e., a transition is defined whenever it satisfies the constraints in h_{YZ} or h_{ZY}. We define $T_{\text{total}} \in \mathcal{T}$ as the largest BTS that includes all possible transitions. Second, we remove all secret-revealing states from T_{total} and define

$$T_0 = T_{\text{total}}|_{Q \setminus Q_{\text{rev}}}.$$

Then, we need to solve a *safety game* by iteratively removing inconsistent states from T until the BTS is consistent. Specifically, we define a removing operator

$$F : \mathcal{T} \to \mathcal{T}$$

by: for any T, we have $F(T) = T|_{Q^T_{\text{consist}}}$. Note that $T|_{Q^T_{\text{consist}}}$ need not be consistent in general since removing inconsistent states may create new inconsistent states. Note that, iterating the removing operator F always converges in a finite number of steps since we need to remove at least one state for each iteration and there are only finite number of states in T. We define

$$T^* = \lim_{k \to \infty} F^k(T_0), \text{ where } T_0 = T_{\text{total}}|_{Q \setminus Q_{\text{rev}}}$$

as the resulting BTS after the convergence of the removing operator F.

Remark 2. (Simplification of information states): In the information state updating rule, at each instant, the current-state estimate is added to the second component. However, if a current-state estimate does not even contain a secret state, it is impossible to infer that the system was at a secret state no matter what is observed in the future. For such a scenario, adding the current-state estimate is irrelevant for us to determine Q_{rev}. Therefore, if $C(\imath') \cap X_S = \emptyset$, then the second part of the information state updating rule in the following can be further simplified as:

$$D(\imath') = \{\widetilde{\text{UR}}_\gamma(\widetilde{\text{NX}}_\sigma(\rho)) \in 2^{X \times X} : \rho \in D(\imath)\}. \tag{15}$$

This simplification does not affect any results of the proposed approach but can reduce the space state significantly when the number of secret states is relatively small. For the sake of simplicity, we will use this simplified rule in the example by omitting current-state estimate in the second component when no secret state is involved.

Before we proceed further, we illustrate the above procedure by the following example.

Example 5: Let us still consider system G in Fig. 1(a). Then, T^* in Fig. 2 is a BTS. For the sake of simplicity, uncontrollable events o_1 and o_2 are omitted in each control decision in Fig. 2. Also, we follow the simplification in the above remark by omitting secret-irrelevant information in the second part of Z-states. Furthermore, some decisions are "equivalent" at a Y-state in the sense that some events in the control decision may not be feasible within the unobservable reach. For example, decisions $\{a\}$ and $\{a, c\}$ are equivalent at s_1 in Fig. 2 as event c is not feasible. For those equivalent decisions, we only draw the one with all redundant events included in the figure, which is without loss of generality for the purpose of control synthesis. At the initial Y-state $s_1 = (\{0\}, \{\emptyset\})$, the supervisor makes control decision $\gamma = \{b, c\}$, we reach Z-state $s_5 = h_{YZ}(s_1, \gamma) = (\{0, 4, 5, 7\}, \{\{(0, 0), (0, 4), (0, 5), (0, 7), (4, 4), (4, 5), (4, 7), (5, 5), (7, 7)\}\}, \{b, c\})$. From s_5, the occurrence of observable event o_2 leads to the next Y-state $s_6 = h_{ZY}(s_5, o_2) = (\{6, 8\}, \{\{(0, 6), (0, 8), (4, 6), (4, 8), (5, 6), (7, 8)\}\})$. From Y-state s_6, by making control decision $\gamma = \{a, b, c\}$, the system will reach the next Z-state $s_7 = h_{YZ}(s_6, \gamma) = (\{6, 8\}, \{\{(0, 6), (0, 8), (4, 6), (4, 8), (5, 6), (7, 8)\}, \{(6, 6), (8, 8)\}\}, \{a, b, c\})$ according the original updating rule in (8). Note that since the current-state estimate is $C(\imath') = \{6, 8\}$ in which there is no secret state, we can apply the simplification rule without adding $\{\odot_\gamma(C(\imath'))\}$. This is why we have $D(s_7) = \{\{(0, 6), (0, 8), (4, 6), (4, 8), (5, 6), (7, 8)\}\}$ in Fig. 2. From Z-state s_7, observable event o_1 occurs and the system will reach the next Y-state $s_8 = h_{ZY}(s_7, o_1) = (\{6\}, \{\{(0, 6), (4, 6), (7, 6)\}\})$. Then, supervisor makes control decision $\gamma = \{a, b, c\}$ and system will reach to Z-state $s_9 = h_{YZ}(s_8, \gamma) = (\{6\}, \{\{(0, 6), (4, 6), (7, 6)\}\}, \{a, b, c\})$. Again, $\{\odot_\gamma(\{6\})\}$ is not added as no secret state is involved.

In fact, T^* is obtained as follows. First, we construct the largest BTS that enumerates all possible transitions for each state, which is T_{total} shown in the entire box marked with black lines in Fig. 2. Note that $Q_{\text{rev}} = \{s_{18}\}$ since $D_1(s_{18}) = \{\{0, 1, 2\}, \{5\}\}$ and $\{5\} \subseteq X_S$. Therefore, T_0 is obtained by restricting T_{total} to $Q \setminus Q_{\text{rev}}$, which is shown in the box marked with blue lines in Fig. 2. Then, we need to remove all inconsistent states from T_0. Since s_{18} has been removed, s_{15} becomes an inconsistent Z-state, which should be removed by applying operator F for the first time. Then, s_{12} becomes a new inconsistent Z-state as feasible observable event o_2 is not defined; hence, s_{12} is deleted when applying operator F for the second time. We keep

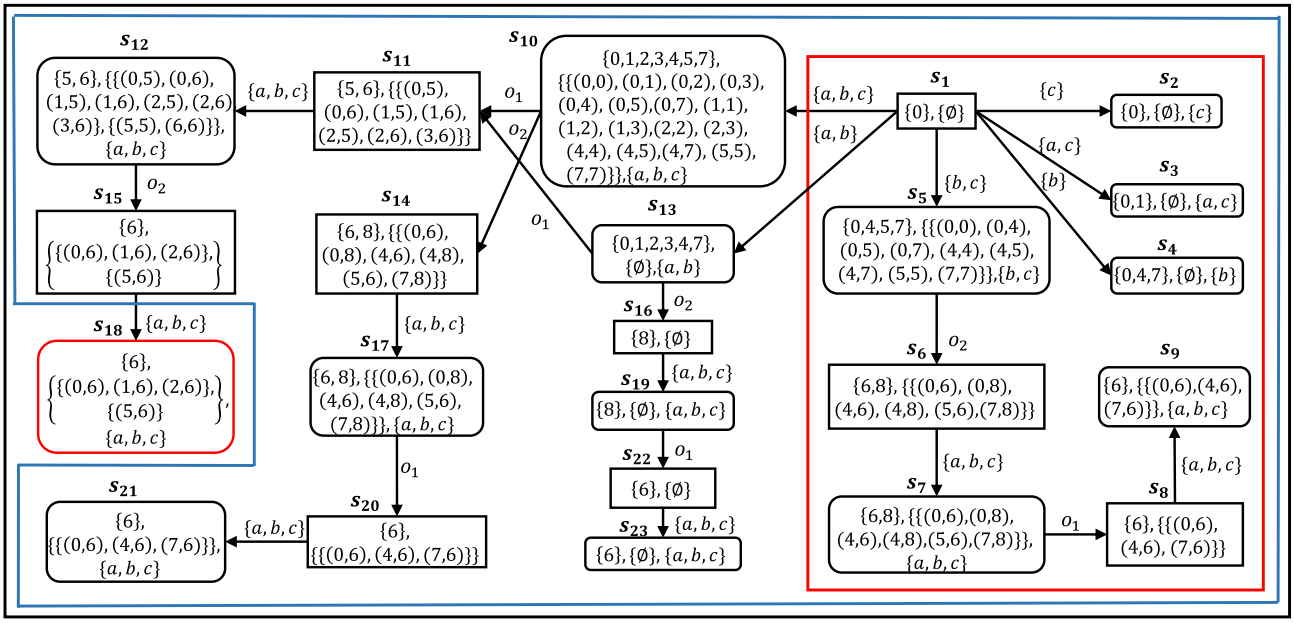


Fig. 2. Example of the construction of the BTS. Rectangular states correspond to Y-states and rectangular with rounded corners states correspond to Z-states. T_{total} is the entire system in the box marked with black lines, which is the largest BTS that enumerates all transitions, T_0 is the system in the box marked with blue lines, and T^* is the system in the box marked with red lines.

applying operator F and need to remove state s_{11}. This makes s_{10} and s_{13} inconsistent because feasible observable event o_1 is not defined; operator F will further remove them. Then, operator F converges to T^*.

Next, we show that synthesizing a supervisor within T^* is without loss of generality.

Theorem 1: The opacity-enforcing synthesis problem has no solution if T^* is empty.

Proof (Sketch): Suppose that the synthesis problem has a solution S. Then, we know that any information states reached in S are not in Q_{rev}. Moreover, for any Y- or Z-states reached by supervisor S are consistent as S can correctly choose a transition. Therefore, all information states reached under S should not be removed during the iteration of operator F. Hence, T^* should not be empty if the synthesis problem has a solution. ■

Then, for any Y-state y in T^*, we define

$$\text{Dec}_{T^*}(y) = \{\gamma \in \Gamma : h_{YZ}^{T^*}(y, \gamma)!\}$$

as the set of control decisions defined at y in T^*. Clearly, $\langle \text{Dec}_{T^*}(y), \subseteq \rangle$ forms a finite poset, which contains at least one maximal element γ such that $\forall \gamma' \in \text{Dec}_{T^*}(y) : \gamma \not\subset \gamma'$. When T^* is not empty, we can synthesize a supervisor S^* as follows. At each instant, the supervisor S^* will remember the current information state y (Y-state) and pick a *maximal* control decision γ from $\text{Dec}_{T^*}(y)$. Note that maximal control decision is not unique in general and we denote by $\text{Dec}_{T^*}^{\max}(y) \subseteq \text{Dec}_{T^*}(y)$ the set of maximal control decisions at y in T^*. Then, we update the information state based on the control decision issued and wait for the next observable event and so forth. The execution of S^* is formally described as Procedure 1.

Procedure 1: Execution of Qualitative Supervisor S^*.

input : T^*
output: control decision of S^* at each instant
$y \leftarrow \{\{x_0\}, \{\emptyset\}\}$;
find a maximal control decision γ from $\text{Dec}_{T^*}^{\max}(y)$;
make initial control decision γ;
$z \leftarrow h_{YZ}^{T^*}(y, \gamma)$;
while *new event* $\sigma \in \gamma \cap \Sigma_o$ *is observed* **do**
  $y \leftarrow h_{ZY}^{T^*}(z, \sigma)$;
  find a maximal control decision γ from $\text{Dec}_{T^*}^{\max}(y)$;
  update the control decision to γ;
  $z \leftarrow h_{YZ}^{T^*}(y, \gamma)$;

Finally, we show that the proposed supervisor S^* indeed solves Problem 1.

Theorem 2: Supervisor S^* executed as Procedure 1 enforces infinite-step opacity and is maximally permissive.

Example 6: Again we consider system G in Fig. 1(a) and we use Procedure 1 to solve Problem 1. At the initial Y-state $y_0 = (\{x_0\}, \{\emptyset\})$, we have $\text{Dec}_{T^*}^{\max}(y_0) = \{\{a, c\}, \{b, c\}\}$. If we choose control decision $\{a, c\}$, then we reach Z-state s_3 and the system has no future observation. This gives supervisor S_1 shown in Fig. 3(a), which results in the language generated by G_1 in Fig. 1(b).

On the other hand, if we choose control decision $\{b, c\}$ initially, then the system moves to s_5. From s_5, the occurrence of o_2 leads to s_6 and the supervisor picks $\{a, b, c\}$ as the maximal control decision and the system moves to s_7. Then, observable events o_1 occurs and the system moves to s_8, where supervisor

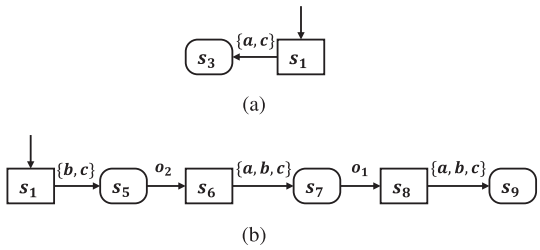


Fig. 3. Solutions S_1 and S_2. (a) Solution S_1. (b) Solution S_2.

S_2 picks $\{a, b, c\}$ leading the system to state s_9. This gives supervisor S_2 shown in Fig. 3(b), which results in the language generated by G_2 in Fig. 1(c). As we have discussed in Example 2, both supervisors are maximal and they are incomparable.

Finally, we discuss the complexity of the proposed qualitative supervisor synthesis algorithm. To synthesis an opacity-enforcing supervisor, first, we need to construct the largest BTS T_{total}, which contains at most $2^{|X|+2^{|X|\times|X|}}$ Y-states and $2^{|\Sigma_c|+|X|+2^{|X|\times|X|}}$ Z-states. For each Y-state, there are at most $2^{|\Sigma_c|}$ transitions defined, and for each Z-state, there are at most $|\Sigma_o|$ transitions defined. Therefore, in the worst case, T_{total} contains $2^{|X|+2^{|X|\times|X|}} + 2^{|\Sigma_c|+|X|+2^{|X|\times|X|}}$ states and $2^{|X|+2^{|X|\times|X|}} \cdot 2^{|\Sigma_c|} + 2^{|\Sigma_c|+|X|+2^{|X|\times|X|}} \cdot |\Sigma_o|$ transitions. The complexity of removing all secret-revealing states from T_{total} to obtain T_0 is linear in the size of T_{total}. The complexity of removing all inconsistent states iteratively to obtain T^* is quadratic in the size of T_{total}. Next, we use Procedure 1 to execute the qualitative supervisor S^* using T^*. During the execution, the supervisor needs to record the information state. The information state contains at most $|X| \times 2^{|X|\times|X|}$ states. By making a new control decision upon the occurrence of a new observable event, the information state is updated in exponential time in the size of G. Overall, the entire complexity of the proposed synthesis algorithm is doubly exponential in the size of the system G for the offline synthesis strategy, but is single exponential for the online execution stage in Procedure 1.

VI. Quantifying Secret-Revelation-Time

So far, we have solved a qualitative version of privacy-enforcing control problem by requiring that the closed-loop system under control is infinite-step opaque. In some cases, such a binary requirement may be too strong as the secret may be revealed inevitably after some delays no matter what control policy is taken, i.e., infinite-step opacity is not enforceable.

In most of the applications, however, the importance of secret will decrease as time goes on. Then, for the scenario where infinite-step opacity is not enforceable, it makes sense to consider an optimal synthesis problem by maximizing the *secret-revelation-time* for each visit of secret states. In the rest of this article, we will implement this idea by further generalizing the qualitative synthesis problem to a quantitative version by ensuring the secret be revealed *as late as possible* within a finite security-preserving horizon.

A. Problem Formulation of Quantitative Synthesis Problem

Let S be a supervisor, $\alpha \in P(\mathcal{L}(S/G))$ be an observation, and $\alpha' \leq \alpha$ be a prefix of α. We define

$$\text{REV}(\alpha', \alpha) = \{\beta : \alpha' \leq \beta \leq \alpha \wedge \hat{X}_{S/G}(\alpha' \mid \beta) \subseteq X_S\}$$

as the set of observations that are prefixes of α and upon which the visit of secret state at instant α' is revealed. If a secret state is visited at instant α' and it is revealed when α is executed, then we have $\text{REV}(\alpha', \alpha) \neq \emptyset$. Note that once we know that the system was at a secret state at instant α', we know this forever. Therefore, we are interested in the *first instant* when the secret is revealed, i.e., the shortest string $\beta_{\text{short}} \leq \alpha$ in $\text{REV}(\alpha', \alpha)$. Then, $|\beta_{\text{short}}| - |\alpha'|$ is referred to as the *secret-revelation-time* for the instant α' upon α.

To quantify when the visit of a secret state is revealed, we consider a cost function that assigns each secret-revelation-time to a nonnegative cost by

$$\Delta : \mathbb{N} \to \mathbb{R} \tag{16}$$

such that Δ is monotonically nonincreasing and will decrease to zero in $N_{\max}$ steps, where $N_{\max} \in \mathbb{N}$ is a nonnegative integer associated with function Δ. Cost function of the above form is reasonable in most cases because the importance of secret decreases as time goes on and revealing secret early usually yields high cost. For example, in location-based services, the intruder may want to infer whether or not a user has been to some secret locations, e.g., a hospital, which may be related to the health condition of the user. Clearly, knowing the health condition two days ago is more sensitive than knowing the health condition a year ago, because the former is more related to the current health condition of the user. Note that here we consider parameter $N_{\max}$ as a security-preserving horizon associated with the cost function, i.e., after $N_{\max}$ steps the secret is free to be revealed.

To avoid counting the revelation of each visit of secret state duplicatively, we define the cost incurred for α' upon α as the cost incurred at the *first* secret-revelation-instant as we assume the cost function is nonincreasing, i.e.,

$$\texttt{Cost}(\alpha', \alpha) = \begin{cases} 0, & \text{if} \quad \text{REV}(\alpha', \alpha) = \emptyset \\ \Delta(|\beta_{\text{short}}| - |\alpha'|), & \text{if} \quad \text{REV}(\alpha', \alpha) \neq \emptyset \end{cases}$$

where β_{short} is the shortest string in $\text{REV}(\alpha', \alpha)$. Note that, for string $st \in \mathcal{L}(S/G)$, if $\delta(s) \notin X_S$, then we always have $\texttt{Cost}(P(s), P(st)) = 0$.

Finally, we define the total cost incurred along an observation. Note that there may have multiple visits of secret states along string α at different instants. Therefore, we define the *total cost* incurred upon $\alpha \in P(\mathcal{L}(S/G))$ as the summation of the costs for all visits of secret states and their associated secret-revelation-times, i.e.,

$$\texttt{Cost}(\alpha) = \sum_{\alpha' \leq \alpha} \texttt{Cost}(\alpha', \alpha).$$

Then, the cost of the closed-loop system S/G, i.e.,

$$\texttt{Cost}(S/G) = \max_{\alpha \in P(\mathcal{L}(S/G))} \texttt{Cost}(\alpha).$$

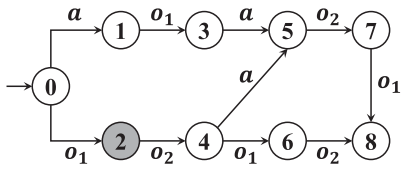


Fig. 4. System G with $\Sigma_o = \{o_1, o_2\}, \Sigma_c = \{a\}$, and $X_S = \{2\}$.

Then, the quantitative privacy-enforcing synthesis problem is then formulated as follows.

Problem 2: (Quantitative privacy-enforcing control problem) Given a system G, a set of secret states $X_S \subseteq X$, and a cost function $\Delta : \mathbb{N} \to \mathbb{R}$ associated with upper bound $N_{\max}$, determine whether or not there exists a supervisor $S : P(\mathcal{L}(G)) \to \Gamma$ such that its cost is finite. If so, synthesize an optimal supervisor S such that the following holds:

1) for any S', we have $\texttt{Cost}(S/G) \leq \texttt{Cost}(S'/G)$;
2) for any S' such that $\texttt{Cost}(S/G) = \texttt{Cost}(S'/G)$, we have $\mathcal{L}(S/G) \not\subset \mathcal{L}(S'/G)$.

Intuitively, the cost of a closed-loop system is defined as the *worst-case cost* and the control problem is essentially a min–max optimization problem aiming to minimize the worst-case cost of the system under control. Note that, we require the cost of the synthesized supervisor to be finite in order to avoid the case of repeatedly revealing secrets. Specifically, if there is an uncontrollable sequence of events along which secret can be revealed repeatedly, then we cannot find a supervisor such that the cost of closed-loop system is finite. Clearly, such a supervisor leading to infinite cost is meaningless and we consider this case as no solution.

Remark 3: The quantitative formulation using cost function can also captures the notions of current-state opacity [51] and K-step opacity [52] in the literature, which requires that the visit of a secret state should not be revealed currently and in the next K steps, respectively. Specifically, K-step opacity, where $K \in \mathbb{N}$ is an integer, it suffices to consider cost function Δ_{Kst} defined by

$$\forall k \leq K : \Delta_{Kst}(k) = \infty \text{ and } \forall k > K : \Delta_{Kst}(k) = 0$$

and current-state opacity is nothing but 0-step opacity. For the above designed cost function, we observe that $N_{\max} = K$. These existing notions are essentially binary, while our new formulation allows to investigate the effect of secret revelation delays more quantitatively.

Remark 4: For the sake of simplicity, hereafter, we consider a cost function Δ in a simple specific form of

$$\Delta(k) = \max\{N_{\max} - k, 0\},$$

where $N_{\max}$ is a finite value. That is, $\Delta(k)$ is the cost incurred if the visit of a secret state is revealed for the first time after k steps. Our approach, in principal, can be applied to any form of cost function as long as it decreases to zero in a finite number of steps.

Example 7: Let us consider system G in Fig. 4 with $\Sigma_o = \Sigma_{uc} = \{o_1, o_2\}$ and $X_S = \{2\}$. Note that all events in string $o_1o_2o_1o_2$ are uncontrollable and there does not exist another string in G having the same projection. Therefore, we cannot enforce infinite-step opacity qualitatively, i.e., no matter what the supervisor does, the intruder will know for sure that the system was at secret state 2 for the instant when $o_1o_2o_1o_2$ is observed. However, the supervisor can control how late the secret is revealed.

For example, let us consider supervisor S_1 that always enables all events. Then, for $P(o_1o_2o_1o_2) = o_1o_2o_1o_2$, we have $\textsc{Rev}(o_1, o_1o_2o_1o_2) = \{o_1o_2o_1o_2\}$ and $\textsc{Rev}(\alpha', o_1o_2o_1o_2) = \emptyset$ for $\alpha' \neq o_1$. Therefore, we have

$$\texttt{Cost}(o_1o_2o_1o_2) = \sum_{\alpha' \leq o_1o_2o_1o_2} \texttt{Cost}(\alpha', o_1o_2o_1o_2)$$

$$= \texttt{Cost}(o_1, o_1o_2o_1o_2) = \Delta(|o_1o_2o_1o_2| - |o_1|) = N_{\max} - 3$$

Similarly, we can compute $\texttt{Cost}(o_1o_2o_2o_1) = \texttt{Cost}(o_1, o_1o_2o_2o_1) = \Delta(|o_1o_2o_2| - |o_1|) = N_{\max} - 2$, which is the worst-case cost in S_1/G, i.e., $\texttt{Cost}(S_1/G) = N_{\max} - 2$.

If we consider supervisor S_2 that disables a initially, then we have $\textsc{Rev}(o_1, o_1o_2o_1o_2) = \{o_1, o_1o_2, o_1o_2o_1, o_1o_2o_1o_2\}$ and $\textsc{Rev}(o_1, o_1o_2o_2o_1) = \{o_1, o_1o_2, o_1o_2o_2, o_1o_2o_2o_1\}$. Therefore, $\texttt{Cost}(o_1o_2o_1o_2) = \texttt{Cost}(o_1o_2o_2o_1) = \texttt{Cost}(o_1) = N_{\max}$, which is the worst-case cost in S_2/G, i.e., $\texttt{Cost}(S_2/G) = N_{\max}$.

B. Augmented Information State

To solve the quantitative synthesis problem, the previous proposed information state is not sufficient as the time information is lost, i.e., we only remember all possible delayed-state estimates without specifying *when* they are visited. Therefore, we further augment this information to the previous proposed information states, which leads to the *augmented information states* defined by

$$I_a := 2^X \times 2^{2^{X \times X} \times \{0,1,\ldots,N_{\max}-1\}}.$$

Each augmented information state $\imath \in I_a$ is in the form of $\imath = (C_a(\imath), D_a(\imath))$. The first component $C_a(\imath) \in 2^X$ is still a current-state estimate. Each element in the second component $D_a(\imath)$ is in the form of

$$(\rho, k) = (\{(x_1, x_1'), (x_2, x_2'), \ldots, (x_n, x_n')\}, k) \in D_a(\imath),$$

where the first and the second parts represent, respectively, a delayed-state estimate and when it is visited, and we define

$$\hat{\rho} = \{x_1, x_2, \ldots, x_n\} = \{x \in X : (x, x') \in \rho\}.$$

Note that we only augment the timing information for $N_{\max}$ steps as we consider a cost function that decreases to zero in $N_{\max}$ steps. For an augmented information state $\imath \in I_a$, define

$$\text{SR}(\imath) = \{(\rho, k) \in D_a(\imath) : \hat{\rho} \subseteq X_S\} \tag{17}$$

as those secret-revealing delayed-state estimates.

Similarly, suppose that the system's augmented information state is $\imath = (C_a(\imath), D_a(\imath))$. Then, upon the occurrence of an observable event $\sigma \in \Sigma_o$ and a new control decision $\gamma \in \Gamma$, the augmented information state $\imath$ is updated to $\imath' = (C_a(\imath'), D_a(\imath'))$ as follows:

Augmented information state updating rule

$$\begin{cases} C_a(\imath') = \mathrm{UR}_\gamma(\mathrm{NX}_\sigma(C_a(\imath))) \\ D_a(\imath') = \left\{ \begin{matrix} (\widetilde{\mathrm{UR}}_\gamma(\widetilde{\mathrm{NX}}_\sigma(\rho)), k+1) : \\ (\rho, k) \in D_a(\imath) \backslash \mathrm{SR}(\imath), k+1 < N_{\max} \end{matrix} \right\} \\ \qquad \cup \{(\odot_\gamma(C_a(\imath')), 0)\}. \end{cases} \tag{18}$$

Compared with the information state updating rule in (8), the augmented information state updating rule in (18) has the following differences.

1) The timing information is also tracked, which increases a time unit upon the occurrence of each observable event.
2) We only update the delayed-state estimates in the last $N_{\max} - 2$ steps with which the secrets have not yet been revealed. This is because the cost decreases to zero within $N_{\max}$ steps and we only consider the cost incurred for the first secret-revelation-instant.

Still, let $\alpha = \sigma_1\sigma_2\ldots\sigma_n \in P(\mathcal{L}(S/G))$ be an observable string. We denote by $\mathcal{I}_a(\alpha)$ the augmented information state reached by α, which is defined according to (9) using the augmented updating rule in (18) with initial state $\imath_0 = (\mathrm{UR}_{S(\epsilon)}(\{x_0\}), \{(\odot_{S(\epsilon)}(\mathrm{UR}_{S(\epsilon)}(\{x_0\})), 0)\})$. Similar to the previous information state, the augmented information state has the following properties.

Proposition 2: Let S be a supervisor, $\alpha \in P(\mathcal{L}(S/G))$ be an observable string, and $\mathcal{I}_a(\alpha)$ be the augmented information state reached. Then, we have the following:

1) $C_a(\mathcal{I}_a(\alpha)) = \hat{X}_{S/G}(\alpha)$;
2) $D_a(\mathcal{I}_a(\alpha)) =$
$$\left\{ (\rho_{\alpha',\alpha}, |\alpha \mid -|\alpha' \mid) : \begin{matrix} \alpha' \le \alpha, |\alpha \mid -|\alpha' \mid < N_{\max}, \\ [\forall \alpha' \le \beta < \alpha : \mathrm{Rev}(\alpha', \beta) = \emptyset] \end{matrix} \right\}.$$

Proof: The proof is very similar to that of Proposition 1, hence, a detailed proof is omitted. The only differences are: 1) the augmented updating rule has a counter that remembers the number of steps between α' and α, and 2) only those delayed-state estimates that are not secret revealing are updated, i.e., $\mathrm{Rev}(\alpha', \beta)$ should be empty set for any strict prefix β before α, otherwise, the delayed-state estimate will be dropped according to the updating rule. ■

With the help of Proposition 2, now we can relate the proposed augmented information state with the cost function as follows. Note that a secret-revelation cost occurs at the instant when the secret is revealed *for the first time*. This is captured by $\mathrm{SR}(\mathcal{I}_a(\alpha))$ and for any $(\rho, k) \in \mathrm{SR}(\mathcal{I}_a(\alpha))$ such that $\hat{\rho} \subseteq X_S$, this secret revelation is only counted once as it will not be updated according to the augmented updating rule. Therefore, we can define a state-based cost function

$$\Delta_I : I_a \to \{0, 1, \ldots, N'_{\max}\}$$

assigning each augmented information state $\imath \in I_a$ a cost by

$$\Delta_I(\imath) = \begin{cases} \sum_{(\rho,k)\in \mathrm{SR}(\imath)} \Delta(k), & \text{if } \mathrm{SR}(\imath) \neq \emptyset \\ 0, & \text{if } \mathrm{SR}(\imath) = \emptyset. \end{cases} \tag{19}$$

Note that the cost of each information state is upper bounded by $N'_{\max} \le 1 + 2 + \cdots + N_{\max} = \frac{1}{2} N_{\max}(N_{\max}+1)$, which corresponds to the extreme case where each of the previous $N_{\max}$ steps visits a secret state and they are all revealed for the first time at the current instant.

The following result shows that, for any observable string $\alpha \in P(\mathcal{L}(S/G))$, the total secret-revelation cost incurred $\mathtt{Cost}(\alpha)$ is equal to the summation of the costs of all information states reached along α.

Theorem 3: Let S be a supervisor, $\alpha \in P(\mathcal{L}(S/G))$ be an observable string, and for each prefix $\alpha' \le \alpha$, $\mathcal{I}_a(\alpha')$ be the augmented information state reached by α'. Then, we have

$$\mathtt{Cost}(\alpha) = \sum_{\alpha' \le \alpha} \Delta_I(\mathcal{I}_a(\alpha')). \tag{20}$$

Proof: Suppose $\alpha = \sigma_1\sigma_2\ldots\sigma_n \in P(\mathcal{L}(S/G))$ be an observable string. Let $i_1 < i_2 < \cdots < i_k$ be the indices such that $\mathrm{Rev}(\sigma_1\ldots\sigma_{i_p}, \alpha) \neq \emptyset\, \forall p = 1, \ldots, k$. Then, for each i_p, we denote by j_p the first instant when the secret at instant $\sigma_1\ldots\sigma_{i_p}$ revealed, i.e., $\sigma_1\ldots\sigma_{j_p}$ is the shortest string in $\mathrm{Rev}(\sigma_1\ldots\sigma_{i_p}, \alpha)$. Then, by the definition of $\mathtt{Cost}(\alpha)$, we have

$$\mathtt{Cost}(\alpha) = \sum_{\alpha' \le \alpha} \mathtt{Cost}(\alpha', \alpha) = \sum_{p=1,\ldots,k} \Delta(j_p - i_p).$$

Then, let $\imath_0, \imath_1, \ldots, \imath_n$ be all augmented information states reached along α. Then, by (18) and Proposition 2, each $p = 1, \ldots, n$ only contributes a cost of $\Delta(j_p - i_p)$ via Δ_I at information state $\imath_{j_p}$. Therefore, we have

$$\begin{aligned} \sum_{\alpha' \le \alpha} \Delta_I(\mathcal{I}_a(\alpha')) &= \sum_{\alpha' \le \alpha} \sum_{(\rho,k)\in \mathrm{SR}(\mathcal{I}_a(\alpha'))} \Delta(k) \\ &= \sum_{i=1,\ldots,n} \Delta_I(\imath_i) = \sum_{p=1,\ldots,k} \Delta(j_p - i_p). \end{aligned}$$

This completes the proof. ■

Example 8: Let us consider system G in Fig. 4 and $N_{\max} = 5$. Suppose that supervisor S enables a initially, i.e., $S(\epsilon) = \{o_1, o_2, a\}$. Then, we have

$$\begin{aligned} \mathcal{I}_a(\epsilon) &= (\mathrm{UR}_{S(\epsilon)}(\{x_0\}), \{(\odot_{S(\epsilon)}(\mathrm{UR}_{S(\epsilon)}(\{x_0\})), 0)\}) \\ &= (\{0,1\}, \{(\{(0,0),(0,1),(1,1)\}, 0)\}). \end{aligned}$$

When event o_1 is observed, if the decision of the supervisor is $S(o_1) = \{o_1, o_2\}$, then the augmented information state is updated to $\mathcal{I}_a(o_1)$, where

$$C_a(\mathcal{I}_a(o_1)) = \mathrm{UR}_{S(o_1)}(\mathrm{NX}_{o_1}(C_a(\mathcal{I}_a(\epsilon)))) = \{2, 3\},$$

$$\begin{aligned} D_a(\mathcal{I}_a(o_1)) &= \left\{ \begin{matrix} (\widetilde{\mathrm{UR}}_{S(o_1)}(\widetilde{\mathrm{NX}}_{o_1}(\rho)), k+1) : \\ (\rho, k) \in D_a(\mathcal{I}_a(\epsilon)) \setminus \mathrm{SR}(\mathcal{I}_a(\epsilon)), k+1 < 5 \end{matrix} \right\} \\ &\quad \cup \{(\odot_{S(o_1)}(C_a(\mathcal{I}_a(o_1))), 0)\} \\ &= \left\{ \begin{matrix} (\{(0,2),(0,3),(1,3)\}, 1), \\ (\{(2,2),(3,3)\}, 0) \end{matrix} \right\}. \end{aligned}$$

Again, if the supervisor further makes control decision $S(o_1o_2) = \{o_1, o_2, a\}$ when event o_2 is observed, then, the

augmented information state is updated to $\mathcal{I}_a(o_1o_2)$, where

$$\begin{aligned} &C_a(\mathcal{I}_a(o_1o_2)) = \mathrm{UR}_{S(o_1o_2)}(\mathrm{NX}_{o_2}(C_a(\mathcal{I}_a(o_1)))) = \{4,5\},\\ &D_a(\mathcal{I}_a(o_1o_2))\\ &= \left\{\begin{array}{c}(\widetilde{\mathrm{UR}}_{S(o_1o_2)}(\widetilde{\mathrm{NX}}_{o_2}(\rho)),k+1):\\ (\rho,k)\in D_a(\mathcal{I}_a(o_1))\backslash \mathrm{SR}(\mathcal{I}_a(o_1))),k+1<5\end{array}\right\}\\ &\cup\{(\odot_{S(o_1o_2)}(C_a(\mathcal{I}_a(o_1o_2))),0)\}\\ &= \left\{\begin{array}{c}(\{(0,4),(0,5)\},2),(\{(2,4),(2,5)\},1),\\ (\{(4,4),(4,5),(5,5)\},0)\end{array}\right\}. \end{aligned}$$

Note that there is a secret-revealing delayed-state estimate in $\mathcal{I}_a(o_1o_2)$. Specifically, for $(\rho,k)=(\{(2,4),(2,5)\},1)\in D_a(\mathcal{I}_a(o_1o_2))$, we have $\hat{\rho}=\{2\}\subseteq X_S$. Therefore, we have $\mathrm{SR}(\mathcal{I}_a(o_1o_2))=\{(\{(2,4),(2,5)\},1)\}$ and the cost of augmented information state is $\Delta_I(\mathcal{I}_a(o_1o_2))=\Delta(1)=N_{\max}-1=4$.

C. Quantitative Synthesis Algorithm

Theorem 3 suggests the basic idea for solving the quantitative synthesis problem. One can consider the privacy-enforcing control problem as an optimal control problem for *accumulated total cost*. Similar to the safety control problem over the information state space for the qualitative synthesis, we can solve the quantitative optimal control problem over the augmented state space defined by the *augmented bipartite transition system* (A-BTS), which is the same of the BTS but incorporating the augmented information- state updating rule.

Definition 3: An A-BTS T w.r.t. G is a 7-tuple

$$T=\left(Q_Y^T,Q_Z^T,h_{YZ}^T,h_{ZY}^T,\Sigma_o,\Gamma,y_0^T\right) \tag{21}$$

where the following holds:

1) $Q_Y^T\subseteq I_a$ is the set of Y-states. Therefore, a Y-state $y\in Q_Y^T$ is in the form of $y=(C_a(y),D_a(y))$;
2) $Q_Z^T\subseteq I_a\times\Gamma$ is the set of Z-states. For each $z\in Q_Z^T$, and $I_a(z)$ and $\Gamma(z)$ denote, respectively, the augmented information state and the control decision, so that $z=(I_a(z),\Gamma(z))$. For simplicity, we write $z=(C_a(z),D_a(z),\Gamma(z))$;
3) $h_{YZ}^T:Q_Y^T\times\Gamma\to Q_Z^T$ is the partial transition function from Y-states to Z-states satisfying the following constraint: for any $h_{YZ}^T(y,\gamma)=z$, we have

$$\begin{cases} C_a(z)=\mathrm{UR}_\gamma(C_a(y))\\ D_a(z)=\{(\widetilde{\mathrm{UR}}_\gamma(\rho),k):(\rho,k)\in D_a(y)\}\\ \qquad\cup\{(\odot_\gamma(C_a(z)),0)\}\\ \Gamma(z)=\gamma\end{cases} \tag{22}$$

4) $h_{ZY}^T:Q_Z^T\times\Sigma\to Q_Y^T$ is the partial transition function from Z-states to Y-states satisfying the following constraint: for any $h_{ZY}^T(z,\sigma)=y$, we have $\sigma\in\Gamma(z)\cap\Sigma_o$ and

$$\begin{cases} C_a(y)=\mathrm{NX}_\sigma(C_a(z))\\ D_a(y)=\left\{\begin{array}{c}(\widetilde{\mathrm{NX}}_\sigma(\rho),k+1):\\ (\rho,k)\in D_a(z)\backslash\mathrm{SR}(z),k+1<N_{\max}\end{array}\right\}\end{cases} \tag{23}$$

5) Σ_o is the set of observable events of G;
6) Γ is the set of admissible control decisions of G;
7) $y_0^T:=(\{x_0\},\{\emptyset\})\in Q_Y^T$ is the initial Y-state.

To solve the quantitative synthesis problem, one can still think it as a two-player game between the supervisor and the environment. The goal of the supervisor is to minimize the total cost incurred, while the environment wants to maximize the cost. Still, we construct the largest A-BTS w.r.t. G that enumerates all the feasible transitions satisfying the constraints of h_{YZ}^T and h_{ZY}^T and denote such an all-feasible A-BTS by T_{total}, which is the arena of the game. For each state $q\in Q_Y^{T_{\text{total}}}\cup Q_Z^{T_{\text{total}}}$, we denote by $\mathrm{POST}(q)$ the set of all its successor states.

In order to synthesize an optimal strategy, we need to compute the best worst-case cost one can guarantee starting from each state, which is referred to as the *value* of the state. Specifically, the value of each state can be computed by the following value iterations:

$$V_{k+1}(q)=\begin{cases}\min_{q'\in\mathrm{POST}(q)}V_k(q'), & \text{if } q\in Q_Y^{T_{\text{total}}}\\ \max_{q'\in\mathrm{POST}(q)}V_k(q')+\Delta_I(\mathcal{I}_a(q)), & \text{if } q\in Q_Z^{T_{\text{total}}}\end{cases} \tag{24}$$

and the initial value function is

$$\forall q\in Q_Y^{T_{\text{total}}}\cup Q_Z^{T_{\text{total}}}:V_0(q)=0.$$

The above is the standard value iteration technique that has been extensively investigated in the literature, either in the context of optimal total-cost control problem [53], [54] or in the context of resource games [55], [56]. Intuitively, for each Y-state, since we are able to pick one control decision, the value of this state is the minimum of the values of all its successor states. However, for each Z-state, since we need to consider all possible observations, the value of this state is the maximum of the values of all its successor states. Note that, by our construction, the secret-revelation cost occurs at each Z-state z such that $\mathrm{SR}(I_a(z))\neq\emptyset$; this is why the instant cost is also added to Z-states at each iteration.

Although the value iteration will converge to the value function denoted by V^*, it may take infinite steps as the value of some states may be infinite. This is because there may exist a cycle in the A-BTS structure such that: 1) an instant cost will incur in the cycle, and 2) one cannot avoid this cycle by choosing control decisions at Y-states. This case corresponds to the scenario where the system repeatedly visits secret states and reveals the secret. In this case, the values of states in the cycle will keep increasing for each round of iteration, i.e., $V_0(q)<V_1(q)<V_2(q)<\cdots$ and the value of state q is actually infinity, i.e., $V^*(q)=\infty$. However, it is known that such a value for each state can be determined only by a finite number of iterations for at most $L=n^2\cdot N'_{\max}$ steps [57], where $n=|Q_Y^{T_{\text{total}}}\cup Q_Z^{T_{\text{total}}}|$ is the number of states in T_{total}. Specifically, by computing value function V_L, we have

$$V^*(q)=\begin{cases}V_L(q), & \text{if } V_L(q)<n\cdot N'_{\max}\\ \infty, & \text{otherwise.}\end{cases} \tag{25}$$

In other words, $V^*(q)$ is the best cost-to-go the supervisor can guarantee at state q.

Based on the above discussion, Procedure 2 is proposed to solve Problem 2. First, it builds T_{total} based on the A-BTS and

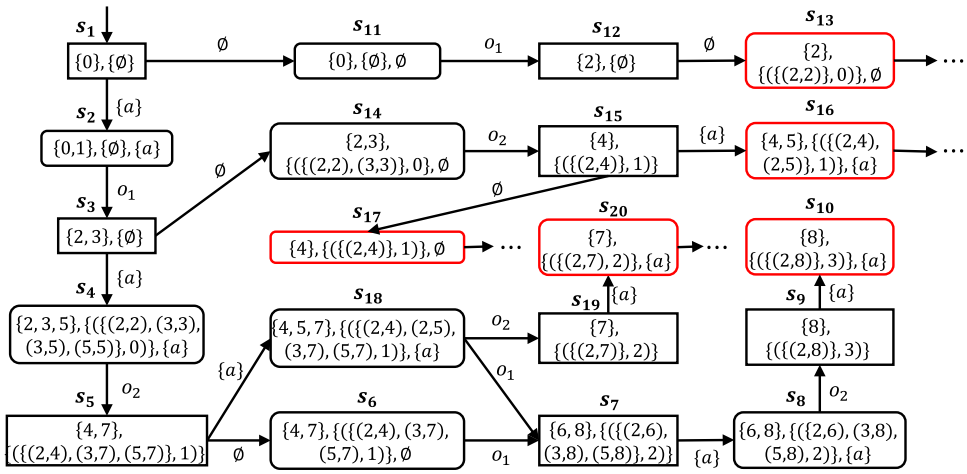


Fig. 5. Example of the construction of the A-BTS T_{total}. Rectangular states correspond to Y-states and rectangular states with rounded corners correspond to Z-states.

Procedure 2: Execution of Quantitative Supervisor S^*.

input : V^*
output: control decision of S^* at each instant
$y \leftarrow \{\{x_0\},\{\emptyset\}\}$, $\Delta_{\text{rem}} = V^*(y)$;
find a control decision γ from $\text{Dec}^{\max}_{V^*}(y, \Delta_{\text{rem}})$;
make initial control decision γ;
$z \leftarrow h_{YZ}(y,\gamma)$, $\Delta_{\text{rem}} \leftarrow \Delta_{\text{rem}} - \Delta_I(\mathcal{I}_a(z))$;
while *new event* $\sigma \in \gamma \cap \Sigma_o$ *is observed* **do**
  $y \leftarrow h_{ZY}(z,\sigma)$;
  find a control decision γ from $\text{Dec}^{\max}_{V^*}(y, \Delta_{\text{rem}})$;
  update the control decision to γ;
  $z \leftarrow h_{YZ}(y,\gamma)$, $\Delta_{\text{rem}} \leftarrow \Delta_{\text{rem}} - \Delta_I(\mathcal{I}_a(z))$;

computes the value function V^*. The optimal value of a state represents the best worst-case cost one can guarantee starting from this state. Therefore, the optimal value of the initial Y-state is the best cost-to-go that the supervisor can guarantee for the closed-loop system. If $V^*(y_0) = \infty$, then we cannot find a supervisor whose cost is finite. Otherwise, $V^*(y_0)$ is the optimal cost one can achieve, i.e., $\texttt{Cost}(S/G) = V^*(y_0)$. To execute the supervisor, we use a variable Δ_{rem} to record the total cost remained for the supervisor to attain the optimal value. Formally, let V^* be the value function and y be a Y-state in T_{total}, and Δ_{rem} be the cost remaining. We define

$$\text{Dec}_{V^*}(y, \Delta_{\text{rem}}) = \{\gamma \in \Gamma : z = h^{T_{\text{total}}}_{YZ}(y,\gamma), V^*(z) \leq \Delta_{\text{rem}}\}$$

as the set of all control decisions that attain the optimal value. Clearly, $\langle \text{Dec}_{V^*}(y, \Delta_{\text{rem}}), \subseteq \rangle$ is also a finite poset and we also denote by $\text{Dec}^{\max}_{V^*}(y, \Delta_{\text{rem}})$ the set of all maximal elements in $\text{Dec}_{V^*}(y, \Delta_{\text{rem}})$. At each Y-state, the supervisor chooses a maximal control decision from $\text{Dec}^{\max}_{V^*}(y, \Delta_{\text{rem}})$. Once a Z-state is reached, the cost remained is updated to $\Delta_{\text{rem}} - \Delta_I(\mathcal{I}_a(z))$ as a state cost incurred.

Theorem 4: Supervisor S^* executed as Procedure 2 solves Problem 2.

Remark 5: Still, in the augmented information-state updating rule, when $C_a(\imath') \cap X_S = \emptyset$, we do not really need to add $\{(\odot_\gamma(C_a(\imath')), 0)\}$ to $D_a(\imath')$. This is because the estimate of such an instant will never contribute to the cost function Δ_I no matter what is observed in the future. For the sake of clarity, we used the original completed rule in (18) for the theoretical developments. However, for the sake of simplicity, we will adopt this simplification in the following illustrative example. The reader should be aware of this discrepancy.

Finally, we illustrate how to synthesize an optimal quantitative supervisor by the following example.

Example 9: Let us consider system G in Fig. 4 and our goal is to synthesize an optimal supervisor such the secret-revelation cost of the closed-loop system is minimized. Suppose that $N_{\max} = 5$. First, we construct the largest A-BTS T_{total} that enumerates all possible transitions, which is partially shown in Fig. 5. For each Z-state in T_{total}, we find those secret-revealing delayed-state estimates and assign each of them a state cost. Specifically, we have $\Delta_I(s_{10}) = 2$, $\Delta_I(s_{13}) = 5$, $\Delta_I(s_{16}) = 4$, $\Delta_I(s_{17}) = 4$, and $\Delta_I(s_{20}) = 3$ and all the other states have zero cost. For example, for state $s_{10} = \{\{8\},\{(\{(2,8)\},3)\},\{a\}\}$, we have $\Delta_I(s_{10}) = \Delta(3) = 5 - 3 = 2$ as $\text{SR}(s_{10}) = \{(\{(2,8)\},3)\}$. For this example, since there is only one secret state and the system is acyclic, we omit all successor states from s_{13}, s_{16}, s_{17}, and s_{20} as these states do not contribute to the cost/value function.

Next, we iteratively update the value of each state in T_{total} by value iterations. The value iteration procedure is given in Table I, which converges in ten steps. Then, the value of the initial Y-state $V^*(s_1) = 2$ is the optimal cost that we can achieve. Then, we use Procedure 2 to solve Problem 2. The resulting supervisor is shown in Fig. 6 . At the initial Y-state, we have $\Delta_{\text{rem}} = V^*(s_1) = 2$ and $\text{Dec}^{\max}_{V^*}(s_1, 2) = \text{Dec}_{V^*}(s_1, 2) = \{\{a\}\}$. Then, we choose control decision $\{a\}$ and move to Z-state s_2. Note that cost Δ_{rem} remains unchanged as $\Delta_I(s_2) = 0$. Then, observable event o_1 occurs and the system moves to s_3. At state s_3, we have

TABLE I
VALUE ITERATIONS OF T_{TOTAL}

$V_i(q)$	0	1	2	3	...	10	11
s_1	0	0	0	0	...	2	2
s_2	0	0	0	0	...	2	2
s_3	0	0	0	0	...	2	2
s_4	0	0	0	0	...	2	2
s_5	0	0	0	0	...	2	2
s_6	0	0	0	0	...	2	2
s_7	0	0	0	0	...	2	2
s_8	0	0	0	2	...	2	2
s_9	0	0	2	2	...	2	2
s_{10}	0	2	2	2	...	2	2
s_{11}	0	0	0	5	...	5	5
s_{12}	0	0	5	5	...	5	5
s_{13}	0	5	5	5	...	5	5
s_{14}	0	0	0	4	...	4	4
s_{15}	0	0	4	4	...	4	4
s_{16}	0	4	4	4	...	4	4
s_{17}	0	4	4	4	...	4	4
s_{18}	0	0	0	3	...	3	3
s_{19}	0	0	3	3	...	3	3
s_{20}	0	3	3	3	...	3	3

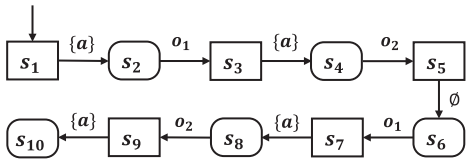


Fig. 6. Solution S.

$\text{Dec}^{\max}_{V^*}(s_3, 2) = \{\{a\}\}$. By choosing control decision $\{a\}$, we reach s_4 and we still have $\Delta_{\text{rem}} = 2 - \Delta_I(s_4) = 2$. Then, the system moves to s_5 upon the occurrence of event o_2. At state s_5, we have $\text{Dec}^{\max}_{V^*}(s_5, 2) = \{\emptyset\}$, then by choosing $\emptyset$ we reach s_6. We repeat the above process, and obtain the supervisor shown in Fig. 6, which is an optimal and maximally permissive supervisor.

The complexity analysis of the proposed quantitative supervisor synthesis algorithm is similar to the qualitative counterpart. First, we need to build the largest A-BTS T_{total}, In the worst case, T_{total} contains $2^{|X|+2^{|X|\times|X|}\times N_{\max}} + 2^{|X|+2^{|X|\times|X|}\times N_{\max}+|\Sigma_c|}$ states and $2^{|X|+2^{|X|\times|X|}\times N_{\max}} \cdot 2^{|\Sigma_c|} + 2^{|X|+2^{|X|\times|X|}\times N_{\max}+|\Sigma_c|} \cdot |\Sigma_o|$ transitions. Then, we compute the value function V^* based on T_{total}, whose complexity is exponential in the size of T_{total} in the worst case. Therefore, the theoretical complexity for the offline synthesis part is triple exponential in the size of G. However, once V^* is computed the online execution of Procedure 2 is still single exponential in the size of G since it essentially requires to update the augmented information-state on-the-fly.

VII. CONCLUSION

In this article, we systematically investigate both qualitative and quantitative synthesis of privacy-enforcing supervisors based on the notion of infinite-step opacity. For the qualitative case, we define a new class of BTSs that captures the delayed information in the control synthesis problem over a game structure. Based on the BTS, we proposed an effective algorithm that solves the standard infinite-step opacity control problem without the assumption that all controllable events are observable, which is restrictive and required by the existing work. For the quantitative case, we propose the notion of secret-revelation-time as a quantitative measure for infinite-step opacity. By suitably augmenting the timing information into the BTS, we solve the quantitative synthesis problem as an optimal total-cost control problem. Note that, the quantitative part essentially solves a min–max optimization problem by considering the worst-case cost. In the future, we also plan to consider stochastic models, such as Markov decision processes, and to minimize the expectation of the secret-revelation cost.

APPENDIX A
PROOF OF PROPOSITION 1

Proof: The first component $C(\mathcal{I}(\alpha))$ is obtained by iteratively applying UR and NX operators, which gives (i) according to [28]. Next, we prove (ii) by induction on the length of α.

Induction Basis: Suppose that $|\alpha| = 0$, i.e., $\alpha = \epsilon$. Then, we know that $\mathcal{I}(\epsilon) = \imath_0$. Let $\gamma = S(\epsilon)$ and we have

$$\begin{aligned}
&D(\mathcal{I}(\epsilon))\\
&=\{\odot_\gamma(\text{UR}_\gamma(\{x_0\}))\}\\
&=\left\{\left\{(x,\delta(x,w))\in X\times X:\ \ x\in\text{UR}_\gamma(\{x_0\}), w\in(\Sigma_{uo}\cap\gamma)^*\right\}\right\}\\
&=\left\{\left\{(\delta(s),\delta(st))\in X\times X:\ \ st\in(\Sigma_{uo}\cap\gamma)^*\ \right\}\right\}\\
&=\{\{(\delta(s),\delta(st))\in X\times X: st\in\mathcal{L}(S/G), P(s)=P(st)=\epsilon\}\}\\
&=\{\rho_{\epsilon,\epsilon}\}. \qquad (26)
\end{aligned}$$

That is, the induction basis holds.

Induction Step: Now, let us assume that (ii) holds for $|\alpha| = k$. Then, we want to prove the case of $\alpha\sigma$, where $\sigma \in \Sigma_o \cap \gamma$. In the following equations, $\gamma = S(\alpha)$. By the updating rule in (8) and (i), we know that

$$\begin{aligned}
D(\mathcal{I}(\alpha\sigma)) = &\left\{\widetilde{\text{UR}}_\gamma(\widetilde{\text{NX}}_\sigma(\rho))\in 2^{X\times X}:\rho\in D(\mathcal{I}(\alpha))\right\}\\
&\cup\left\{\odot_\gamma\left(\hat{X}_{S/G}(\alpha\sigma)\right)\right\}. \qquad (27)
\end{aligned}$$

Since $|\alpha| = k$, by the induction hypothesis, we know

$$D(\mathcal{I}(\alpha)) = \{\rho_{\alpha',\alpha} \in 2^{X\times X} : \alpha' \le \alpha\}.$$

Recall that

$$\rho_{\alpha',\alpha} = \left\{(\delta(s),\delta(st))\in X\times X:\ \begin{array}{c} st\in\mathcal{L}(S/G),\\ P(s)=\alpha', P(st)=\alpha\end{array}\right\}.$$

Therefore, we have

$$\begin{aligned}
&\widetilde{\text{UR}}_\gamma(\widetilde{\text{NX}}_\sigma(\rho_{\alpha',\alpha}))\\
&=\{(x_1,\delta(x_2,\sigma w))\in X\times X:(x_1,x_2)\in\rho_{\alpha',\alpha}, w\in(\Sigma_{uo}\cap\gamma)^*\}\\
&=\left\{\begin{array}{c}(\delta(s),\delta(st\sigma w))\\ \in X\times X\end{array}:\ \begin{array}{c} st\in\mathcal{L}(S/G), w\in(\Sigma_{uo}\cap\gamma)^*,\\ P(s)=\alpha', P(st)=\alpha\end{array}\right\}\\
&=\left\{(\delta(s),\delta(st'))\in X\times X:\ \begin{array}{c} st'\in\mathcal{L}(S/G),\\ P(s)=\alpha', P(st')=\alpha\sigma\end{array}\right\}
\end{aligned}$$

$= \rho_{\alpha',\alpha\sigma}$.

This further gives

$$\begin{aligned}&\left\{\widetilde{\mathrm{UR}}_\gamma(\widetilde{\mathrm{NX}}_\sigma(\rho)) \in 2^{X\times X} : \rho \in D(\mathcal{I}(\alpha))\right\}\\ &= \left\{\widetilde{\mathrm{UR}}_\gamma(\widetilde{\mathrm{NX}}_\sigma(\rho_{\alpha',\alpha})) \in 2^{X\times X} : \alpha' < \alpha\sigma\right\}\\ &= \left\{\rho_{\alpha',\alpha\sigma} \in 2^{X\times X} : \alpha' < \alpha\sigma\right\}. \end{aligned} \tag{28}$$

Moreover, we have

$$\begin{aligned}&\odot_\gamma\left(\hat{X}_{S/G}(\alpha\sigma)\right)\\ &= \left\{(x,\delta(x,w)) \in X\times X : x \in \hat{X}_{S/G}(\alpha\sigma), w \in (\Sigma_{uo}\cap\gamma)^*\right\}\\ &= \left\{(\delta(s),\delta(sw)) \in X\times X : \begin{array}{c} s\in\mathcal{L}(S/G),\\ w\in(\Sigma_{uo}\cap\gamma)^*, P(s)=\alpha\sigma\end{array}\right\}\\ &= \left\{(\delta(s),\delta(sw)) \in X\times X : \begin{array}{c} sw\in\mathcal{L}(S/G),\\ P(s)=P(sw)=\alpha\sigma\end{array}\right\}\\ &= \rho_{\alpha\sigma,\alpha\sigma}. \end{aligned} \tag{29}$$

Therefore, by combining (28) and (29) we have

$$\begin{aligned}D(\mathcal{I}(\alpha\sigma)) &= \{\rho_{\alpha',\alpha\sigma} \in 2^{X\times X} : \alpha' < \alpha\sigma\} \cup \{\rho_{\alpha\sigma,\alpha\sigma}\}\\ &= \{\rho_{\alpha',\alpha\sigma} \in 2^{X\times X} : \alpha' \le \alpha\sigma\}. \end{aligned} \tag{30}$$

This completes the induction step, i.e., (ii) holds. ■

Appendix B
Proof of Theorem 2

Proof: Let $\alpha \in P(\mathcal{L}(S^*/G))$ be any observable string in the closed-loop system S^*/G. By Corollary 1, we know that $D_1(\mathcal{I}_{S^*/G}(\alpha)) = \{\hat{X}_{S^*/G}(\alpha' \mid \alpha) : \alpha' \le \alpha\}$. Since all secret-revealing states have been removed from T_{total}, all information states in T^* are safe, which implies that $\hat{X}_{S^*/G}(\alpha' \mid \alpha) \not\subseteq X_s$. This means that S^* enforces infinite-step opacity.

Next, we show that S^* is maximal. Assume that there exists another supervisor S' such that the closed-loop system S'/G is infinite-step opaque and $\mathcal{L}(S^*/G) \subset \mathcal{L}(S'/G)$. This means that there exists a string $w \in \mathcal{L}(S^*/G) \subset \mathcal{L}(S'/G)$ such that $S^*(w) \subset S'(w)$ and $S^*(w') = S'(w')\ \forall w' < w$. Then, Y-state reached upon the occurrence of string w under supervisors S' and S^* is the same, which is denoted by y. Since S' is a solution to Problem 1, its control decision $S'(w)$ should not be removed at y during the iteration. This means that $S'(w) \in \mathrm{Dec}_{T^*}(y)$, which further gives $S'(w) \in \mathrm{Dec}^{\max}_{T^*}(y)$ and $S^*(w) \notin \mathrm{Dec}^{\max}_{T^*}(y)$. However, it contradicts to our choice in Procedure 1 (line 7) that $S^*(w)$ is in $\mathrm{Dec}^{\max}_{T^*}(y)$. Hence, no such S' exists. ■

Appendix C
Proof of Theorem 4

Proof: The largest A-BTS T_{total} enumerates all feasible control decisions of G. The value iterations in T_{total} guarantees that $V^*(y_0)$ is the best cost-to-go starting from the initial state. Suppose that there exists a supervisor S' such that $\mathtt{Cost}(S'/G) < \mathtt{Cost}(S^*/G)$. Then, the value of the initial Y-state should be at least $\mathtt{Cost}(S'/G)$, which is a contradiction to the result of the value iteration.

The proof of maximal permissiveness is similar to that of Theorem 2. Since we choose a control decision from $\mathrm{Dec}^{\max}_{V^*}(y, \Delta_{\text{rem}})$, which is the set of maximal control decisions that attain optimal value at state y. Any other more permissive choices will result in a supervisor whose worst-case cost is larger than $\mathtt{Cost}(S^*/G)$, which will violate the optimality. ■

References

[1] J. W. Bryans, M. Koutny, L. Mazaré, and P. Ryan, "Opacity generalised to transition systems," *Int. J. Inf. Secur.*, vol. 7, no. 6, pp. 421–435, 2008.

[2] B. Bérard, K. Chatterjee, and N. Sznajder, "Probabilistic opacity for Markov decision processes," *Inf. Process. Lett.*, vol. 115, no. 1, pp. 52–59, 2015.

[3] C. Keroglou and C. N. Hadjicostis, "Probabilistic system opacity in discrete event systems," *Discrete Event Dyn. Syst.*, vol. 28, pp. 289–314, 2018.

[4] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Verification of state-based opacity using petri nets," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2823–2837, Jun. 2017.

[5] J. Chen, M. Ibrahim, and R. Kumar, "Quantification of secrecy in partially observed stochastic discrete event systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 185–195, Jan. 2017.

[6] K. Zhang and M. Zamani, "Infinite-step opacity of nondeterministic finite transition systems: A bisimulation relation approach," in *Proc. IEEE 56th Conf. Decis. Control*, 2017, pp. 5615–5619.

[7] F. Basile and G. De Tommasi, "An algebraic characterization of language-based opacity in labeled petri nets," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 329–336, 2018.

[8] M. Noori-Hosseini, B. Lennartson, and C. N. Hadjicostis, "Compositional visible bisimulation abstraction applied to opacity verification," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 434–441, 2018.

[9] X. Cong, M. P. Fanti, A. M. Mangini, and Z. Li, "On-line verification of current-state opacity by Petri nets and integer linear programming," *Automatica*, vol. 94, pp. 205–213, 2018.

[10] S. Liu and M. Zamani, "Verification of approximate opacity via barrier certificates," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1369–1374, Oct. 2021.

[11] I. Saadaoui, Z. Li, and N. Wu, "Current-state opacity modelling and verification in partially observed petri nets," *Automatica*, vol. 116, 2020, Art. no. 108907.

[12] S. Mohajerani and S. Lafortune, "Transforming opacity verification to nonblocking verification in modular systems," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1739–1746, Apr. 2020.

[13] D. Lefebvre and C. N. Hadjicostis, "Privacy and safety analysis of timed stochastic discrete event systems using Markovian trajectory-observers," *Discrete Event Dyn. Syst.*, vol. 30, pp. 413–440, 2020.

[14] S. Mohajerani, Y. Ji, and S. Lafortune, "Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement," *IEEE Trans. Autom. Control*, vol. 65, no. 8, pp. 3349–3364, Aug. 2020.

[15] F. Lin, W. Chen, W. Wang, and F. Wang, "Information control in networked discrete event systems and its application to battery management systems," *Discrete Event Dynamic Syst.*, vol. 30, pp. 243–268, 2020.

[16] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annu. Rev. Control*, vol. 41, pp. 135–146, 2016.

[17] S. Lafortune, F. Lin, and C. N. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annu. Rev. Control*, vol. 45, pp. 257–266, 2018.

[18] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

[19] A. Saboori and C. N. Hadjicostis, "Verification of infinite-step opacity and complexity considerations," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1265–1269, May 2012.

[20] Y. Falcone and H. Marchand, "Enforcement and validation (at runtime) of various notions of opacity," *Discrete Event Dyn. Syst.*, vol. 25, no. 4, pp. 531–570, 2015.

[21] X. Yin and S. Lafortune, "A new approach for the verification of infinite-step and K-step opacity using two-way observers," *Automatica*, vol. 80, pp. 162–171, 2017.

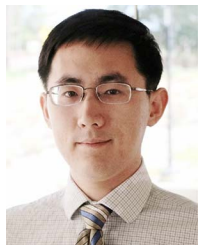
[22] H. Lan, Y. Tong, and C. Seatzu, "Verification of infinite-step opacity using labeled Petri nets," in *Proc. IFAC World Congr.*, 2020, pp. 1729–1734.
[23] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent secrets," *Discrete Event Dyn. Syst.*, vol. 17, no. 4, pp. 425–446, 2007.
[24] S. Takai and Y. Oka, "A formula for the supremal controllable and opaque sublanguage arising in supervisory control," *SICE J. Control, Meas. Syst. Integration*, vol. 1, no. 4, pp. 307–311, 2008.
[25] M. Ben-Kalefa and F. Lin, "Supervisory control for opacity of discrete event systems," in *Proc. IEEE 49th Annu. Allerton Conf. Commun., Control, Comput.*, 2011, pp. 1113–1119.
[26] P. Darondeau, H. Marchand, and L. Ricker, "Enforcing opacity of regular predicates on modal transition systems," *Discrete Event Dyn. Syst.*, vol. 25, no. 1/2, pp. 251–270, 2014.
[27] A. Partovi, T. Jung, and L. Hai, "Opacity of discrete event systems with active intruder," 2020, *arXiv:2007.14960*.
[28] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2140–2154, Aug. 2016.
[29] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Current-state opacity enforcement in discrete event systems under incomparable observations," *Discrete Event Dyn. Syst.*, vol. 28, no. 2, pp. 161–182, 2018.
[30] J. Dubreil, P. Darondeau, and H. Marchand, "Supervisory control for opacity," *IEEE Trans. Autom. Control*, vol. 55, no. 5, pp. 1089–1100, May 2010.
[31] G. Zinck, L. Ricker, H. Marchand, and L. Hélouët, "Enforcing opacity in modular systems," in *Proc. IFAC World Congr.*, 2020, pp. 2157–2164.
[32] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods Syst. Des.*, vol. 40, no. 1, pp. 88–115, 2012.
[33] B. Zhang, S. Shu, and F. Lin, "Maximum information release while ensuring opacity in discrete event systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 4, pp. 1067–1079, Jul. 2015.
[34] B. Behinaein, F. Lin, and K. Rudie, "Optimal information release for mixed opacity in discrete-event systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 4, pp. 1960–1970, Oct. 2019.
[35] X. Yin and S. Li, "Synthesis of dynamic masks for infinite-step opacity," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1429–1441, Apr. 2020.
[36] B. Wu, J. Dai, and H. Lin, "Synthesis of insertion functions to enforce decentralized and joint opacity properties of discrete-event systems," in *Proc. Amer. Control Conf.*, 2018, pp. 3026–3031.
[37] B. Wu and H. Lin, "Privacy verification and enforcement via belief abstraction," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 815–820, Oct. 2018.
[38] Y. Ji, Y.-C. Wu, and S. Lafortune, "Enforcement of opacity by public and private insertion functions," *Automatica*, vol. 93, pp. 369–378, 2018.
[39] Y. Ji, X. Yin, and S. Lafortune, "Enforcing opacity by insertion functions under multiple energy constraints," *Automatica*, vol. 108, 2019, Art. no. 108476.
[40] R. Liu, L. Mei, and J. Lu, "K-memory-embedded insertion mechanism for opacity enforcement," *Syst. Control Lett.*, vol. 145, 2020, Art. no. 104785.
[41] Y. Xie, X. Yin, and S. Li, "Opacity enforcing supervisory control using non-deterministic supervisors," in *Proc. IFAC World Congr.*, 2020, pp. 1763–1769.
[42] A. Saboori and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies via state estimator constructions," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1155–1165, May 2012.
[43] A. Saboori and C. N. Hadjicostis, "Current-state opacity formulations in probabilistic finite automata," *IEEE Trans. Autom. Control*, vol. 59, no. 1, pp. 120–133, Jan. 2014.
[44] X. Yin, Z. Li, W. Wang, and S. Li, "Infinite-step opacity and K-step opacity of stochastic discrete-event systems," *Automatica*, vol. 99, pp. 266–274, 2019.
[45] W. Deng, D. Qiu, and J. Yang, "Fuzzy infinite-step opacity measure of discrete event systems and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 3, pp. 885–892, Mar. 2022.
[46] S. Liu, X. Yin, and M. Zamani, "On a notion of approximate opacity for discrete-time stochastic control systems," in *Proc. Amer. Control Conf.*, 2020, pp. 5413–5418.
[47] D. Lefebvre and C. N. Hadjicostis, "Exposure and revelation times as a measure of opacity in timed stochastic discrete event systems," *IEEE Trans. Autom. Control*, vol. 66, no. 12, pp. 5802–5815, Dec. 2021.
[48] Y. Xie and X. Yin, "Supervisory control of discrete-event systems for infinite-step opacity," in *Proc. Amer. Control Conf.*, 2020, pp. 3665–3671.
[49] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. 2nd ed. Berlin, Germany: Springer, 2008.
[50] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially-observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 5, pp. 1239–1254, May 2016.
[51] Y.-C. Wu and S. Lafortune, "Comparative analysis of related notions of opacity in centralized and coordinated architectures," *Discrete Event Dyn. Syst.*, vol. 23, no. 3, pp. 307–339, 2013.
[52] A. Saboori and C. N. Hadjicostis, "Verification of k-step opacity and analysis of its complexity," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 549–559, Jul. 2011.
[53] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[54] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 1995.
[55] A. Chakrabarti, L. De Alfaro, T. A. Henzinger, and M. Stoelinga, "Resource interfaces," in *Proc. Int. Workshop Embedded Softw.*, 2003, pp. 117–133.
[56] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman, "Temporal specifications with accumulative values," *ACM Trans. Comput. Log.*, vol. 15, no. 4, pp. 1–25, 2014.
[57] Y.-C. Wu and S. Lafortune, "Synthesis of optimal insertion functions for opacity enforcement," *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 571–584, Mar. 2016.

Yifan Xie (Student Member, IEEE) was born in Hubei, China, in 1999. She received the B.Eng. degree in automation from Beihang University, Beijing, China, in 2019, and the M.S. degree in control engineering from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 2022. She is currently working toward the Ph.D. degree in mechanical engineering with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Stuttgart, Germany, and with the International Max Plank Research School for Intelligent System.

Her current research interests include data-driven control, model predictive control, formal methods, and discrete-event systems.

Shaoyuan Li (Senior Member, IEEE) was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree in automatic control theory and application from Nankai University, Tianjin, in 1997.

Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a Professor. His current research interests include model predictive control, dynamic system optimization, and cyber-physical systems.



Xiang Yin (Member, IEEE) was born in Anhui, China, in 1991. He received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2012, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 2013 and 2017, respectively, all in electrical engineering.

Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently an Associate Professor. His research interests include formal methods, discrete-event systems, and cyber-physical systems.

Dr. Yin is the Chair of the IEEE CSS Technical Committee on Discrete Event Systems, an Associate Editor for the *Journal of Discrete Event Dynamic Systems: Theory and Applications*, and a Member of IEEE CSS Conference Editorial Board. He was the recipient of *IEEE Conference on Decision and Control* Best Student Paper Award Finalist in 2016.